HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Yläkoulun ohjelmoinnin opetuksen kehittäminen oppimateriaalin keinoin

Helsingin yliopisto
Matemaattis-luonnontieteellinen tiedekunta
Matematiikan ja tilastotieteen osasto
Matematiikan aineenopettajalinja
Maisterintutkielma

helmikuu 2020
Eeva Haapakangas
Ohjaaja: Mika Koskenoja

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

MATEMAATTIS-LUONNONTIETEELLINEN TIEDEKUNTA
MATEMATISK-NATURVETENSKAPLIGA FAKULTETEN
FACULTY OF SCIENCE

Tiedekunta – Fakultet – Faculty	Koulutusohjelma – Utbildningsprogram – Degree programme
Matemaattis-luonnontieteellinen tiedekunta	Matematiikan aineenopettajalinja

Tekijä – Författare – Author

Eeva Haapakangas

Työn nimi – Arbetets titel – Title

Yläkoulun ohjelmoinnin opetuksen kehittäminen oppimateriaalin keinoin

Työn laji – Arbetets art – Level	Aika – Datum – Month and year	Sivumäärä – Sidoantal – Number of pages
Maisterintutkielma	helmikuu 2020	100 + 2

Tiivistelmä – Referat – Abstract

Vuoden 2016 opetussuunnitelmassa ohjelmointi tuotiin uutena alueena yläkouluihin osaksi matematiikan opetusta. Tämän tutkielman tarkoituksena on selvittää, millaisia erilaisia oppimateriaaleja tässä opetuksessa käytetään ja miten yläkoulun ohjelmoinnin opetusta voitaisiin kehittää oppimateriaalien avulla. Oppimateriaaliksi tässä tutkielmassa määritellään oppikirja (digitaalinen tai paperinen), oppi-/tehtäväkirja (digitaalinen tai paperinen), tehtäväkirja (digitaalinen tai paperinen), opettajan materiaali, verkkopohjaiset oppimisympäristöt (voidaan toteuttaa eri teknologioilla, kuten pilvipalveluna tai verkkoympäristössä), muut teknologiaympäristöt, kuten esimerkiksi opetuskäyttöön suunnitellut ohjelmoitavat robotit, elektroniikka-alustat, älypuhelinten ohjelmointi ja pelit.

Erityisesti etäopiskelun yhteydessä käytetyt ja kehitetyt teknologiat vaikuttavat myös ohjelmoinnin opetuksessa käytettäviin oppimateriaaleihin. Elektroninen oppiminen, mobiilioppiminen ja ubiikki oppiminen muovaavat tulevaisuudessa myös ohjelmoinnin opetuksessa käytettäviä materiaaleja ja ympäristöjä.

Tutkielman osana tehtiin tutkimustehtävä, jossa kysyttiin yläkoulun opettajilta heidän kokemuksiaan ohjelmoinnin opetuksesta sekä parannusehdotuksia erityisesti oppimateriaaleihin. Tutkimus sisälsi sekä monivalinta- että avoimia kysymyksiä ohjelmointikielistä ja -ympäristöistä, koulujen teknologiaympäristöistä ja oppimateriaaleista. Myös opettajien omia kokemuksia ohjelmoinnin opetuksesta kysyttiin. Tutkimuksen aineisto koostui 34 matematiikan opettajan vastauksista.

Tämän tutkimuksen tuloksia voi hyödyntää ennen kaikkea suunniteltaessa oppimateriaalia ohjelmoinnin opetukseen yläkouluihin. Yhteenvetona ohjelmoinnin oppimateriaalin kehittämiselle ehdotetaan seuraavat asiat:

1. Käytössä tekstipohjainen ohjelmointikieli, mieluimmin Python.
2. Ohjelmointiympäristö, joka sisältää tuen tehtävien automaattiselle palautukselle ja tarkistamiselle.
3. Harjoitustehtäviä, jotka sisältävät muutakin kuin koodausta, esimerkiksi koodin lukemista, korjaamista, selittämistä ja parantamista.
4. Opetettavaan aiheeseen integroituja harjoitustehtäviä.
5. Eriyttämisen mahdollistavia harjoitustehtäviä.
6. Opettajan materiaalia, joka sisältää tunneilla läpikäytävän aineiston sekä tuntisuunnitelmat.
7. Mahdollisesti erillinen ohjelmoinnin oppimateriaali (oppikirja).

Avainsanat – Nyckelord – Keywords

oppimateriaalit, ohjelmointi, yläkoulu

Säilytyspaikka – Förvaringställe – Where deposited

Helsingin yliopisto

Muita tietoja – Övriga uppgifter – Additional information

Sisällys

1 Johdanto

EU-komissaari Androulla Vassiliou kirjoitti EU:n opetusministereille kirjeen heinäkuussa 2014 kehottaen heitä edistämään lasten mahdollisuuksia kehittää tietokoneen koodaustaitoja koulussa. Vassiliou kirjoittaa koodauksen olevan tärkeää, koska "koodaus on nykypäivän lukutaito. Jokaista vuorovaikutusta tietokoneen kanssa ohjaa koodi. Ohjelmointi on olennaista hyperkytketyn (hyper-connected) maailman ymmärtämiselle." (European Commision, 2014).

Puhuttaessa 2000-luvulla tarvittavasta osaamisesta Ornelas Marques ja Marques (2012) korostavat korkean tason osaamisen, kuten kriittisen ajattelun, ongelmanratkaisukyvyn ja itsenäisen ajattelun, tärkeyttä. Heidän mukaansa ohjelmointikieltä opetellessaan oppilaat kehittävät kaikkia näitä kykyjä ja heti ongelman kohdatessaan heidän täytyy (tuntevat tarvetta) ratkaista se päästäkseen projektissa eteenpäin. Ohjelmointia pidetään tärkeänä kompetenssina myös ongelmanratkaisukykyjen ja loogisen päättelyn kehittymiselle. Täten ohjelmoinnin integroimista osaksi opetusta kaikilla oppitasoilla pidetäänkin kannattavana (Kalelioglu & Gülbahar, 2014).

Yleisesti ottaen ohjelmointi nähdään hyödyllisenä taitona, mutta sen oppiminen on vaikeaa. Tähän vaikeuteen on löydetty useita syitä, kuten vajavainen ymmärrys siitä miten tietokonemallit toimivat, kyvyttömyys hallita koodin lukemista, seuraamista ja kirjoittamista ja kyvyttömyys ymmärtää ylemmän tason konsepteja, kuten ohjelman suunnittelua. (Selby, 2015, Robins ym., 2003) Hadjerrouit (2008) mainitsee ohjelmoinnin olevan vaikeaa, koska se on ennemminkin taitoa kuin tietokokonaisuus.

Ohjelmointi tuli yläkoulujen opetuksen osaksi vuonna 2016 uuden opetussuunnitelman myötä. Ohjelmoinnin opetus liitettiin osaksi matematiikan opetusta. Opetussuunnitelmassa (OPS) ohjelmoinnin opetuksen tavoitteena vuosiluokilla 7–9 ovat oppilaan ohjaaminen kehittämään omaa algoritmista ajatteluaan ja taitoja soveltaa matematiikan ohjelmoinnin osaamista ongelmien ratkaisussa. Keskeiset matematiikan sisältöalueet, jotka

liittyvät ohjelmoinnin opetukseen, ovat algoritmisen ajattelun syventäminen, hyvien ohjelmointikäytäntöjen harjoittelu ohjelmoimalla sekä itse tehtyjen tai valmiiden tietokoneohjelmien soveltaminen matematiikan opiskelussa. (Opetushallitus, 2014)

Tämän tutkielman tarkoituksena on selvittää, millaisia erilaisia oppimateriaaleja käytetään ohjelmoinnin opetuksessa osana matematiikan opetusta yläkoulussa ja miten ohjelmoinnin opetusta voitaisiin kehittää oppimateriaalien avulla. Tutkielman voi jakaa kolmeen eri alueeseen:

1. Tausta ja oppimateriaalit. Tässä osassa esitellään, miten uuteen opetussuunnitelmaan on sisällytetty ohjelmoinnin opintoja ja millaisia tavoitteita näille opinnoille on asetettu. Tämän jälkeen on määritelty, mitä oppimateriaali käsitteenä tarkoittaa tässä tutkimuksessa. Seuraavaksi on esitelty tällä hetkellä Suomessa yleisimmin käytössä olevia matematiikan oppikirjoja, fokusoiden erityisesti siihen miten ohjelmoinnin opetus on sisällytetty niihin. Tähän osioon on sisällytetty myös yleisesti käytettyjen eri ohjelmointikielten vertailu. Vertailussa on otettu huomioon Suomessakin yleisesti käytössä olevat ohjelmointikielet, kuten Scratch, Racket, Python, JavaScript ja Lego-ohjelmointi. Tämä alue on toteutettu tutkimalla nykyisiä oppimateriaaleja sekä kirjallisuustutkimuksella, johon on kerätty tietoa tutkituista ja kokeilluista ohjelmoinnin oppimateriaaleista. Tämä alue sisältää tutkimuksen kappaleet kolme, neljä ja viisi.
2. Didaktiikka ja teknologia. Kirjallisuustutkimus, jossa on tutkittu ohjelmoinnin opetuksessa käytettyä didaktiikkaa ja teknologiaa. Pääpaino on peruskoulutasoisen ohjelmoinnin opetuksen tutkimuksessa. Tutkimus kattaa sekä Suomessa että muissa maissa, joissa ohjelmointia on jo opetettu vuosiluokilla 7–9, aiheesta tehtyjä tutkimuksia. Tutkimuksessa on teoriaosuus ohjelmoinnin opetuksen didaktiikasta ja erilaisten ajatusmallien opettamisesta. Tässä on pyritty etsimään millaisia erilaisia opetuskäytäntöjä ohjelmoinnin opetuksessa on käytössä ja kokeiltu. Didaktiikan tutkimus käsittää kolme eri aluetta, konstruktivismin, ohjelmoinnin opetuksen ja ajatusmallien opettamisen. Lisäksi on pyritty selvittämään

teknologian käytön vaikutusta ohjelmoinnin opetuksessa tänä päivänä ja mahdollisesti tulevaisuudessa. Tämä alue sisältää tutkimuksen kappaleet kuusi ja seitsemän.

3. Tutkimustehtävä. Tässä osassa on esitelty tehdyn tutkimustehtävän sisältö ja tulokset. Tutkimuksessa selvitettiin, millaisia oppimateriaaleja opettajat toivovat opetuksen tueksi. Lisäksi kartoitettiin opettajien kokemuksia ohjelmoinnin opetuksesta. Tutkimus suoritettiin web-pohjaisena kyselynä pääkaupunkiseudun yläkoulun matematiikan opettajille kevään 2018 aikana. Tutkimuksen lopuksi on pohdintaa kirjallisuustutkimuksen ja kyselytutkimuksen pohjalta. Tässä on esitelty myös ideoita mahdollisista aiheeseen liittyvistä jatkotutkimuksista. Tämä alue sisältää tutkimuksen kappaleet kahdeksan ja yhdeksän.

Tämä tutkimus keskittyy siihen, miten yläkoulujen ohjelmoinnin opetusta voitaisiin kehittää oppimateriaalien avulla. Erilaisten ohjelmoinnissa käytettävien työskentelytapojen, kuten pariohjelmointi, kisällioppiminen tai käänteinen opetus, käyttöä ohjelmoinnin alkeisopetukseen ei ole sisällytetty tähän tutkimukseen.

2 Käsitteet

Tässä luvussa selitetään ja avataan tutkielman seuraavissa luvuissa käytettäviä käsitteitä.

Algoritmi tarkoittaa kuvausta tai ohjetta, mitä seuraamalla voidaan saada ratkaistua jokin ongelma. (Mykkänen & Liukas, 2014)

Algoritminen ajattelu ohjaa purkamaan ongelman osiin, etsimään ongelmaan liittyviä toistuvia sääntöjä, luomaan yksikäsitteisiä toimintaohjeita eli algoritmeja ja yleistämään sekä automatisoimaan ratkaisun.

Erittely (engl. decomposition) on laskennallisen ajattelun osa, joka sisältää ongelman jakamisen pienempiin osiin toiminnallisuuden perusteella (Selby & Woollard, 2013).

Laskennallinen ajattelu on ajatteluprosessi, joka sisältää erittelyn (decomposition), abstraktion (abstraction), algoritmin suunnittelun (algorithm design), yleistämisen (generalization) ja arvioinnin (evaluation) (Selby, 2015).

Ohjelmointi tarkoittaa tietokoneelle tai tietokonetta vastaavalle laitteelle annettavien toimintaohjeiden eli algoritmien laatimista (Opetushallitus, 2016).

Ohjelmointiympäristö on tietokoneohjelma, millä ohjelmoija suunnittelee ja toteuttaa ohjelmiston.

Pariohjelmointi on ohjelmointitapa, jossa kaksi ohjelmoijaa työskentelee vierekkäin yhdellä tietokoneella tehden yhteistyössä ohjelman suunnittelua, algoritmia, koodausta tai testausta. (Williams ym. 2002)

Sähköinen oppimateriaali on tässä tutkielmassa tietokoneella luettavissa oleva oppimateriaali, joka hyödyntää myös videoita, ääntä ja interaktiivisia tehtäviä.

Tekstipohjainen ohjelmointikieli on ohjelmointikieli, jolla ohjelmointi tapahtuu ohjelmointilausekkeita kirjoittamalla.

Visuaalinen ohjelmointikieli muodostuu ohjelmalohkoista. Ohjelmoitaessa visuaalisella ohjelmointikielellä ohjelmointi tehdään graafisessa ohjelmointiympäristössä liittämällä ohjelmalohkoja toisiinsa.

Yhteisöllinen oppiminen (engl. collaborative learning) on oppimistilanne, missä kaksi tai useampi oppilas oppivat tai yrittävät oppia jotain yhdessä (Dillenbourg, 1999).

3 Opetussuunnitelma (OPS 2016)

Ohjelmoinnin opetus tuli Suomessa osaksi yläkoulujen opetusta uuden opetussuunnitelman (OPS) tullessa voimaan vuonna 2016. Ohjelmoinnin opetus on mainittu useissa kohdissa opetussuunnitelmaa. Peruskoulun jälkeen oppilaan tulisi osata ohjelmoida yksinkertaisia ohjelmia. Opetussuunnitelmassa ohjelmointi on liitetty osaksi matematiikan opetusta ja on sisällytetty käsitteellisissä ja tiedonalakohtaisissa tavoitteissa.

Ohjelmointia on tarkoitus opetella ensimmäisistä oppivuosista alkaen, matematiikan tavoitteisiin vuosiluokilla 1–2 sisältyy tutustuminen ohjelmoinnin alkeisiin. Vuosiluokkien 3–6 opetuksen tavoitteena on, että oppilas osaa ohjelmoida toimivan ohjelman graafisessa ohjelmointiympäristössä.

Opetussuunnitelmassa ohjelmoinnin opetuksen tavoitteena vuosiluokilla 7–9 ovat oppilaan ohjaaminen kehittämään omaa algoritmista ajatteluaan ja taitoa soveltaa matematiikan ja ohjelmoinnin osaamista ongelmien ratkaisemiseen. Keskeiset matematiikan sisältöalueet, jotka liittyvät ohjelmoinnin opetukseen, ovat algoritmisen ajattelun syventäminen, hyvien ohjelmointikäytäntöjen harjoittelu ohjelmoimalla sekä itse tehtyjen tai valmiiden tietokoneohjelmien soveltaminen matematiikan opiskelussa. (Opetushallitus, 2014)

Opetussuunnitelma sisältää seuraavat tavoitteet matematiikan opetukselle vuosiluokilla 7–9:

> *T20 ohjata oppilasta kehittämään algoritmista ajatteluaan sekä taitojaan soveltaa matematiikkaa ja ohjelmointia ongelmien ratkaisemiseen.*

Matematiikan tavoitteisiin liittyvät keskeiset sisältöalueet sisältävät:

> *S1 Ajattelun taidot ja menetelmät: Harjoitellaan loogista ajattelua vaativia toimintoja kuten sääntöjen ja riippuvuuksien etsimistä ja esittämistä täsmällisesti. Pohditaan ja määritetään vaihtoehtojen lukumääriä. Vahvistetaan oppilaiden*

päättelykykyä ja taitoa perustella. Harjoitellaan matemaattisen tekstin tulkitsemista ja tuottamista. Tutustutaan todistamisen perusteisiin. Harjoitellaan väitelauseiden totuusarvon päättelyä. Syvennetään algoritmista ajattelua. Ohjelmoidaan ja samalla harjoitellaan hyviä ohjelmointikäytäntöjä. Sovelletaan itse tehtyjä tai valmiita tietokoneohjelmia osana matematiikan opiskelua.

Ohjelmointiin liittyvät matematiikan päättöarvioinnin kriteerit hyvälle osaamiselle (arvosanalle 8) oppimäärän päättyessä:

Algoritminen ajattelu ja ohjelmointitaidot:

Opetuksen tavoitteet: T20 ohjata oppilasta kehittämään algoritmista ajatteluaan sekä taitojaan soveltaa matematiikkaa ja ohjelmointia ongelmien ratkaisemiseen.

Osaaminen: Oppilas osaa soveltaa algoritmisen ajattelun periaatteita ja osaa ohjelmoida yksinkertaisia ohjelmia.

Lisäksi opetussuunnitelmaan kuuluu tieto- ja viestintäteknologian osaaminen, josta on mainittu:

Ohjelmointia harjoitellaan osana eri oppiaineiden opintoja.

Matematiikan oppimisympäristöihin ja työtapoihin liittyvät tavoitteet vuosiluokilla 7–9 ovat seuraavat:

Opetuksen lähtökohdat valitaan oppilaita kiinnostavista aiheista, ilmiöistä ja niihin liittyvistä ongelmista. Konkretia toimii edelleen tärkeänä osana matematiikan opiskelua. Rohkaistaan oppilaita käyttämään ajattelua tukevia piirroksia ja välineitä. Opetuksessa käytetään vaihtelevia työtapoja. Ongelmia matematisoidaan, ratkaistaan ja tulkitaan yksin ja yhdessä. Yhdessä työskennellessä jokainen toimii sekä itsensä että ryhmän hyväksi. Oppimispelit ovat yksi motivoiva työtapa.

Tieto- ja viestintäteknologiaa, kuten taulukkolaskentaa ja dynaamista geometriaohjelmistoa, hyödynnetään opetuksen, oppimisen, tuottamisen, arvioinnin sekä luovuuden välineenä.

4 Oppimateriaalit

Perinteisesti oppimateriaali on koostunut oppikirjoista, harjoitustehtävistä ja opettajan materiaaleista. Nämä on toteutettu kirjoina tai muina paperisina materiaaleina. Teknologian kehittyminen, erityisesti digitalisointi, on vaikuttanut paljon myös käytettäviin oppimateriaaleihin.

Heinonen (2005) määritteli väitöskirjassaan oppimateriaalin “oppiainesta sisältäväksi oppikirjaksi, oppi-/tehtäväkirjaksi, tehtäväkirjaksi, opettajan materiaaliksi tai muuksi oheismateriaaliksi (cd-rom, verkkopohjaiset oppimisympäristöt, videot, oheislukemistot ym.)”. Nämä ovat opetustarkoituksiin tehtyjä oppiainesta sisältäviä materiaaleja.

Verkko-oppiminen on osaltaan vaikuttanut oppimateriaalien kehitykseen. Verkko-oppimisen materiaalien levitys tapahtuu yleensä Internetissä (World Wide Web), intranetissä (organisaation sisäinen suljettu verkkopalvelu) tai extranetissä (organisaation ja asiakkaiden/yhteistyökumppaneiden yhteinen suljettu verkkopalvelu) käyttämällä virtuaalisia oppimisympäristöjä. Tämä on tuonut mukanaan ongelmia ja sitä kautta mahdollisuuksia, joista yksi merkittävimmistä on oppimateriaalien järjestäminen ja jakaminen. Tämän seurauksena onkin yleistynyt tapa jakaa oppimateriaali oppimisobjekteina (learning object) tai oppimiskomponentteina (learning component) (Paulsson & Naeve, 2003). Spalter ja van Dam (2003) määrittelevät oppismisobjektin miksi tahansa materiaaliksi, jota voidaan uudelleen käyttää opetuksessa, kuten tuntisuunnitelma, kuva, video tai osa ohjelmoitua koodia. IEEE (2002) on puolestaan määritellyt oppimisobjektin hyvin laveasti digitaalisena tai ei-digitaalisena kokonaisuutena, jota voidaan käyttää oppimiseen, opetukseen tai harjoitteluun. Friesenin (2001) mukaan oppimisobjekteilla tulisi olla seuraavat ominaisuudet:

- Löydettävissä oleva (discoverable). Objektit sisältävät metadatan, jota käytetään niiden kuvaamiseen ja järjestämiseen.
- Modulaarinen. Objektien tulee toimia kuten “musta laatikko” eli sen sisäisellä toiminnalla ei ole merkitystä vaan sitä voidaan käyttää näkyvien rajapintojen avulla.

- Yhteensopiva (interoperable). Objekti toimii avoimien standardien mukaisesti.

Tässä tutkimuksessa oppimateriaalin määrityksen pohjaksi on otettu Heinosen (2005) määritelmä pienin muutoksin eli oppimateriaaliksi määritellään:

- oppikirja (digitaalinen tai paperinen)
- oppi-/tehtäväkirja (digitaalinen tai paperinen)
- tehtäväkirja (digitaalinen tai paperinen)
- opettajan materiaali
- verkkopohjaiset oppimisympäristöt (voidaan toteuttaa eri teknologioilla, kuten pilvipalveluna tai verkkoympäristössä)
- muut teknologiaympäristöt, kuten esimerkiksi opetuskäyttöön suunnitellut ohjelmoitavat robotit, elektroniikka-alustat, älypuhelinten ohjelmointi ja pelit.

Kukin määritelty osa voi olla oma kokonaisuus tai koostua oppimisobjekteista. Vanhentunut teknologia, kuten cd-rom on jätetty määritelmästä pois, koska tänä päivänä useissa kouluissa oppilaiden käytössä olevat tietotekniikan laitteet eivät sisällä mahdollisuutta cd-romien käyttöön.

Heinosen (2005, s. 132) mukaan hyvällä oppimateriaalilla on seuraavat ominaisuudet:

- on selkeä,
- innostaa ja motivoi,
- auttaa riittävästi eriyttämään,
- tukee monipuolisesti erilaisten opetusmenetelmien käyttöä ja
- helpottaa opettajan työtä (hyvät opettajan oppaat).

Perkkilä ym. (2018) muistuttavat, että oppimateriaali on ennen kaikkea opettajan työkalu eikä sen tule antaa ohjata opetusta liikaa – “oppimateriaali on hyvä renki, mutta huono isäntä.”

4.1 Digitaalinen oppimateriaali

Digitaalisilla oppimateriaaleilla tarkoitetaan digitaalisessa muodossa olevaa aineistoa, joka on kohdistettu tietyn aihepiirin tai aineen opiskelua varten. Näiden oppimateriaalien etuna niiden työstettävyys: opettaja, opiskelija tai opiskelijaryhmä voi koota valmiista materiaaleista omaan tarkoitukseensa sopivan kokonaisuuden, kun tekijänoikeuskysymykset on selvitetty. (Meisalo, Sutinen & Tarhio, 2003)

Digitaalisten oppimateriaalien oletetaan lisääntyvän kouluissa sähköisen ylioppilaskokeen ja ohjelmoinnin tultua osaksi koulumaailmaa. Ohjelmoinnin tultua osaksi yläkoulun matematiikan opetusta on kysyntä sähköisille oppimateriaaleille mahdollisesti lisääntymässä. Digitaaliset oppikirjat saattavat muuttaa perusopetuksen käytäntöjä (Kim & Jung, 2010).

Etelä-Koreassa digitaaliset oppimateriaalit on otettu laajaan käyttöön. Kim & Jung (2010) ovat verranneet painettuja ja digitaalisia oppimateriaaleja pedagogian ja oppimistehokkuuden kannalta katsottuna. Heidän mukaansa digitaalisilla oppikirjoilla on havaittu olevan myönteinen vaikutus mm. opiskelijoiden itseohjautuvaan opiskeluun, tehokkuuteen, tiedon hakemiseen, ongelmanratkaisuun, motivaatioon ja itsereflektioon. Digitaalinen oppimateriaali mahdollistaa oppilaille juuri heidän kykyihinsä ja taitoihinsa räätälöidyn sisällön. Jang (2014) kuvaa digitaalisen oppikirjan olevan tulevaisuuteen suuntautunut oppikirja, joka auttaa opiskelijoita opiskelemaan itseohjautuvasti milloin tahansa ja missä tahansa. Se sisältää perinteisen oppikirjan sisällön lisäksi opiskelun apuvälineitä, tehtäviä, sanakirjoja ja mahdollisuuden muistiinpanojen tekemiselle. Koska digitaalinen oppimateriaali sisältää multimediatoiminnallisuutta, kuten videoita, animaatioita, virtuaalista todellisuutta ja hyperlinkkejä, on se sopiva oppilaille, jotka ovat jo lapsesta saakka tottuneet digitaaliseen ympäristöön. Kang & Everhart (2014) toteavat edelleen digitaalisia oppikirjoja pidettävän käyttäjäkeskeisinä oppimistyökaluina, jotka parantavat oppilaiden luovuutta ongelmanratkaisukykyjä ja luovan ajattelun kykyjä motivoimisen, opiskelumahdollisuuksien parantamisen ja itseohjautuvan opiskelun mahdollistamisen kautta.

Jang (2014) on jaotellut digitaalisen oppikirjan funktiot seuraaviin eri alueisiin: oppimateriaali, oppimisen hallinta, oppimisen tuki ja edistäminen sekä vuorovaikutteisuus ja resurssiyhteys. Taulukossa 1 on kuvattu tarkemmin näiden eri funktioiden sisältöä.

Taulukko 1. Digitaalisen oppikirjan toiminnot (Jang, 2014, Kankaanranta, 2015)

Toiminto		**Kuvaus**
Oppimateriaali	Teksti	Kirjoittaminen, muistiinpanot, navigointi ja sivujen selailu, kirjanmerkit
Oppimisen hallinta	Arviointityöväline	Yhteys arviointityövälineisiin digitaalisen oppikirjan sisä- ja ulkopuolella tarjoaa laajennetut oppimateriaalit oppilaan tason ja ymmärryksen tavoittamiseksi
	Kirjoitustyöväline	Dokumentin luonnostelu, editointi ja tulostus samalla kun editoidaan tekstejä, kuvia, musiikkia ja videoleikkeitä
	Opiskelun hallintajärjestelmä	Oppilaan oppimisprosessia dokumentoivan e-portfolion hallinnointi
Oppimisen tuki ja edistäminen	Multimedia	Kuvat, piirustukset, videoleikkeet, ääni, animaatiot, 3D linkitettyinä sisältöihin tai hyperlinkkeihin
	Tiedonhaku	Muiden kurssien ja kouluasteiden oppikirjojen etsintä
	Lähdemateriaalit	Itsenäisen oppimisen materiaalit
	Hyperlinkit	Erilaiset resurssit linkitettyinä netin kautta helppoon viittaamiseen itseohjautuvassa oppimisessa
	Sanakirja	Ajantasaiset sanastoviittaukset, jotka sisältävät aikaisemmat sanakirjamäärittelyt sekä monia kielikäännöksiä
Vuorovaikutteisuus ja resurssiyhteys	Resurssiyhteys	Yhteys kansalliseen tietovarantoon, yhteys erilaisten poliittisten, taloudellisten, sosiaalisten ja kulttuuristen instituutioiden sisältöihin
	Vuorovaikutteisuus	Vuorovaikutus asiantuntijoiden ja muiden instituutioiden kanssa verkon välityksellä

Digitaalisissa oppimateriaaleissa on myös omat ongelmansa. Mardis ym. (2010) tuovat esille seuraavat haasteet digitaalisten kirjojen käyttöönotossa:

- Piilotetut kustannukset digitaalisten kirjojen käyttöönotossa. Tarvittavien laitteiden ja lisenssien kustannukset sekä tarvittavan infrastruktuurin ja tietoliikenneyhteyksien päivittäminen voivat olla yllättävänkin kalliita.
- Heikentynyt käsityskyky digitaalisia kirjoja käytettäessä. Digitaalisen tekstin lukeminen on erilaista kuin kiinteän tekstin ja vaatii erilaista lukutekniikkaa. Tutkimuksen mukaan oppilaat pitivät digitaalisten kirjojen suurimpana haittana vaikeuksia lukea näytöltä. Lisäksi näytöltä lukeminen väsytti silmiä ja vaikeutti keskittymistä lukemiseen. (Jamali ym., 2009)
- Digitaaliset oppimateriaalit eivät huomioi oppilaita, joilla on näkövaikeuksia. Vaikka monet e-kirjojen lukulaitteet tukevat teksti-puhe-käännösominaisuuksia, niistä puuttuvat valikkojen ja navigoinnin ääniohjaus, äänitettävät muistiinpanot ja kirjanmerkkien lisääminen ääniohjauksella.
- Digitaaliset oppikirjat saattavat kasvattaa opiskelijoiden välisiä sosioekonomisia eroja. Kaikilla oppilailla ei ole samanlaisia mahdollisuuksia hankkia tietotekniikkalaitteita ja -yhteyksiä koteihinsa.
- Riittävät tietoliikenneyhteydet. Kaikki koulut eivät ole samanlaisessa asemassa tekemään tarvittavia teknologiahankintoja, kuten internetpohjaisten oppimateriaalien tietoliikenneyhteyksiltä vaadittavaa nopeutta.
- Opettajat eivät ole valmiita hyödyntämään digitaalisia sisältöjä optimaalisilla tavoilla.
- Digitaaliset oppikirjat eivät ratkaise mahdollisia puutteita oppijaksojen sisällöissä.

4.2 Suomessa käytettäviä matematiikan ja ohjelmoinnin oppimateriaaleja

4.2.1 ViLLE

ViLLE on Turun yliopiston kehittämä web-pohjainen oppimisjärjestelmä. Se on yhteisöllinen koulutusväline, jolla kaikki palveluun laaditut tehtävät voidaan arvioida, niitä voidaan kommentoida ja jakaa kaikkien muiden ViLLE-palvelua käyttävien opettajien

kesken. ViLLE perustuu ensisijaisesti aktiiviseen oppimiseen (active learning). Vaikka ViLLEä voidaan käyttää perinteisenä kurssienhallintajärjestelmänä, sen päätavoitteena on harjoittaa ja kannustaa oppilaita oppimaan tarjoamalla aktiivinen sisältö. (Laakso ym., 2018)

ViLLE sisältää monipuolisesti tehtäviä mm. ohjelmoinnista ja matematiikasta. Näistä aineista löytyy myös valmiiksi laadittuja opintopolkukursseja. Ohjelmoinnin opetukseen yläkouluun ViLLE tarjoaa valmiin ohjelmoinnin peruskurssin Python-ohjelmointikielellä. Peruskurssin suorittaminen kestää 17 viikkoa (kaksi 45min pituista oppituntia viikossa) ja se on suunniteltu pidettäväksi yhden lukukauden aikana. (Kaila ym., 2019)

Kuva 1. ViLLE-oppimisjärjestelmän kurssirakenne. Lähde: oppimisanalytiikka.fi/sites/default/files/documentation/Ohjelmointi/YA/opettajan_kirja_ya.pdf

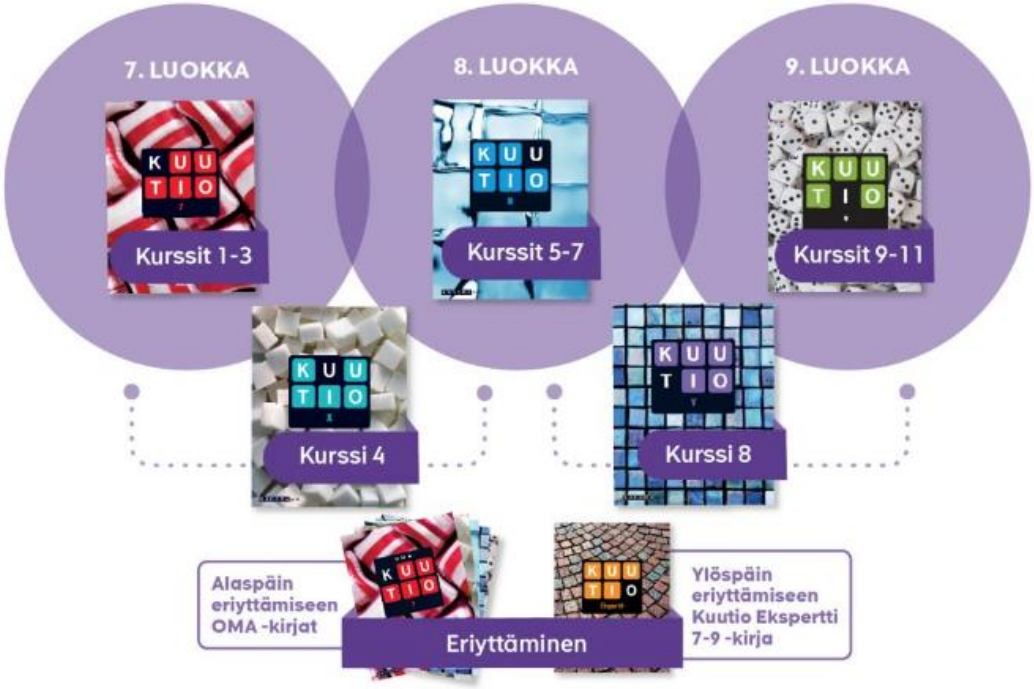
Digitaalinen ViLLE-oppimisjärjestelmä on ollut osana mm. Opetus- ja kulttuuriministeriön rahoittamaa opintopolkujen tutkimusprojektia Lounais-Suomessa. Projektin päämääränä on ollut parantaa opiskelijoiden oppimistuloksia ja motivaatiota ja muodostaa jokaiselle oppilaalle oma henkilökohtainen opintopolku. Turun yliopiston oppimisanalytiikan keskuksella on opetusteknologia-alueella läheistä tutkimusyhteistyötä mm. Salon kaupungin kanssa. (Turun yliopisto, Oppimisanalytiikan keskus, 2020)

4.2.2 Sanoma Pro

Sanoma Pron uuden opetussuunnitelman mukainen matematiikan oppikirjasarja on nimeltään Kuutio. Sarjaan kuuluu kirjat:

- Kuutio 7
- Kuutio 8
- Kuutio 9
- Kuutio X (käsitellään joko 7. tai 8. luokalla, sisältää yläkoulun Potenssi ja polynomi-kurssin)
- Kuutio Y (käsitellään joko 8. tai 9. luokalla, sisältää yläkoulun Funktio ja yhtälöpari-kurssin)
- Kuutio Ekspertti (lisäharjoituskirja ylöspäin eriyttävään opetukseen)

Kirjasarjan jokaisesta kirjasta, poislukien Kuutio Ekspertti, löytyy myös alaspäin eriyttävät OMA-kirjat. Kaikkiin Kuutio-sarjan kirjaan on myös digilisätehtäviä. Python-ohjelmointikokonaisuuksia on jokaiselle luokka-asteelle. Ohjelmointitehtäviä voi tehdä Sanoma Pron Viopen oppimisympäristössä. (Sanoma Pro, 2020a)



Kuva 2. Kuutio-kirjasarja. Lähde: www.sanomapro.fi/sarjat/kuutio/

Sanoma Prolta ilmestyi vuoden 2020 alussa toinen kirjasarja, Muuttuja, yläkoulun matematiikan opetukseen. Kirjasarjaan kuuluu matematiikan käsikirja ja erikseen tehtäväkirjat ja eriyttävät omat työkirjat yläkoulun matematiikan kursseista. (Sanoma Pro, 2020b)

Kuva 3. Muuttuja-kirjasarja. Lähde: www.sanomapro.fi/sarjat/muuttuja/

Sanoma Pro on Suomen suurin oppimateriaalikustantaja noin 40−50% markkinaosuudella (Kilpailu- ja kuluttajavirasto, 2016). Sanoma Pron oppimateriaaleja käyttää 47 000 opettajaa ja yli 3 000 suomalaiskoulussa ja sen digitaalisissa palveluissa on noin miljoona käyttökertaa kuukaudessa (Sanoma Pro, 2020c).

4.2.3 Otava

Otavan matematiikan kirjasarja on Pii. Pii-sarjaan kuuluu kirjat:

- Pii 7
- Pii 8
- Pii 9
- Pii π (voidaan käyttää 7−9. luokalla, sisältää yläkoulun Tilastot ja todennäköisyys- sekä Yhdenmuotoisuus ja trigonometria-kurssin)

Kaikki Pii-kirjasarjan kirjat sisältävät uuden opetussuunnitelman mukaista ohjelmointia. Sarjan digikirjoista ja digiopetusmateriaalista (opettajan materiaali) löytyy ohjelmointiharjoituksia. Jokaiseen kirjaa on saatavilla digilisätehtäviä, jotka auttavat kertaamisessa. (Otava, 2020)

Kuva 4. Pii-kirjasarja. Lähde: oppimisenpalvelut.otava.fi/tuotteet/luokat-7-9/pii-ops-2016/#oppilaan_materiaalit

Otava on Suomen toiseksi suurin oppimateriaalikustantaja noin 30−40% markkinaosuudella (Kilpailu- ja kuluttajavirasto, 2016). Otava julkaisee sisältöjä lähes kaikkiin oppiaineisiin esiopetuksesta lukioon ja tavoittaa vuosittain noin 30 000 opettajaa ja 600 000 oppilasta (Otava, 2020b).

4.2.4 e-Oppi

e-Oppi tarjoaa erilliset oppimateriaalit matematiikan opetukseen ja ohjelmoinnin opetukseen matematiikkaan. Ohjelmointia matematiikkaan -sarjaan kuuluu kolme digikirjaa:

- Ohjelmointia matematiikkaan 7 – Koodauksen alkeet
- Ohjelmointia matematiikkaan 8 – Työkaluja matematiikkaan
- Ohjelmointia matematiikkaan 9 – Taulukot haltuun

Ohjelmointia matematiikkaan -sarjaan on valittu ohjelmointiympäristöksi Processing-ohjelmointi. Processing on Java-kieleen perustuva tekstipohjainen ohjelmointikieli ja -ympäristö, joka on suunniteltu ennen kaikkea kuvien generoimiseen ja muokkaamiseen (Reas & Fry, 2006). Digikirjassa tutustutaan myös Scratch-ohjelmointiympäristöön.

Monista paperisista oppikirjoista on saatavilla digitaalinen versio ja lisämateriaalia digitaalisessa muodossa. Opettajan materiaalit ovat myös saatavilla digitaalisessa muodossa.

Kuva 5. Ohjelmointia matematiikkaan -kirjasarja. Lähde: www.e-oppi.fi/sarja/ohjelmointia-matematiikkaan/

e-Oppi toimii yhteistyössä mm. Jyväskylän yliopiston opettajankoulutuslaitoksen kanssa ja on Peda.net-oppimisalustan pitkäaikaisimpia käyttäjiä. Lisäksi e-Oppi mainitsee yhteistyökumppanikseen Kaarinan kaupungin, jonka kanssa he ovat kehittäneet oppimateriaaleja. (e-Oppi, 2020)

4.2.5 Studeo

Studeon (ent. Tabletkoulu) ohjelmoinnin opetukseen tehty matematiikan digikirjojen lisäksi erillinen digikirja on nimeltään Ohjelmointi. Ohjelmointikieleksi materiaaliin on valittu Python-ohjelmointikieli. Kirja koostuu seitsemästä luvusta. Jokaisessa luvussa ja luvun alaluvussa on harjoitustehtäviä malliratkaisuineen. Opettaja pystyy lisäämään jokaiseen lukuun myös omia tehtäviään. Ensimmäiset neljä lukua käsittelevät ohjelmoinnin perusasioita Python-ohjelmointikielellä. Tämän jälkeen esitellään visuaalinen Turtle-ohjelmointi, funktioiden käyttäminen ohjelmoinnissa sekä opastetaan harjoitustyön tekemisessä.

Kuva 6. Studeon ohjelmoinnin digikirja. Lähde: www.studeo.fi/ylakoulu/oppimateriaalit/

Studeo on sähköisten oppimateriaalien tuottaja. Sen tuotteita käyttää 5 000 opettajaa ja 80 000 oppilasta. (Studeo, 2020)

4.2.6 Edukustannus

Edukustannuksen matematiikan kirjasarja on Luku. Tähän kirjasarjaan kuuluu kirjat:

- Luku 7
- Luku 8
- Luku 9
- tehtävävihkot jokaiselle kirjalla eriyttävään ja itsenäiseen oppimiseen

Luku-kirjasarjassa opettajan materiaali on saatavilla ainoastaan digitaalisessa muodossa. Oppilaiden kirjoja ei ole digitaalisina.

Ohjelmointi Luku-kirjasarjassa on toteutettu algoritmista ajattelua tukemalla Excel-ohjelmointitehtävillä, HTML-kielen perusteilla ja dynaamisen geometriaohjelmiston (GeoGebra) harjoituksilla. (Edukustannus, 2020)

Kuva 7. Luku-kirjasarja. Lähde: www.edukustannus.fi/ylakoulu/luku/

Edukustannus on oppikirjamarkkinoilla verrattain uusi toimija, joka haluaa profiloitua pedagogisesti kunnianhimoisena ja nuorekkaana uudistajana. Edukustannus on osa kolmen tuotemerkin kustantamoa, joka tunnetaan Lasten Keskus ja Kirjapaja Oy:nä. (ePressi, 2016)

4.2.7 Edita

Editan matematiikan kirjasarja on Säde. Kirjasarjaan kuuluu kirjat:

- Säde 1 (käytetään 7. luokalla)
- Säde 2 (käytetään osittain jo 7. luokalla tuntijaosta riippuen, pääosin 8. luokalla)
- Säde 3 (käytetään 8. ja 9. luokalla)
- Säde 4 (käytetään 9. luokalla)

Säde-sarjassa jokaisesta kirjasta erityisopetukseen soveltuva versio. Opettajan materiaali on saatavilla ainoastaan digitaalisessa muodossa. Opettajille on myös koetehtäväpankki. Sarjaan löytyy myös digitehtäviä, joita oppilaat voivat tehdä älypuhelimella, tietokoneella tai tabletilla. Oppikirjoja ei ole digitaalisina.

Ohjelmointi on Säde-sarjassa opettajan materiaalissa. Sieltä löytyvät opetuskokonaisuudet Scratch- sekä Python-ohjelmointiin. Ohjelmointitehtävät tehdään Säde-sarjan omalla ohjelmointialustalla. (Edita, 2020)

Kuva 8. Säde-kirjasarja. Lähde: www.editapublishing.fi/oppimateriaalit/perusopetus/kirjasarjat/sade

5 Ohjelmoinnin opetuksessa käytettävät ohjelmointikielet ja -ympäristöt

Kouluissa tapahtuvan ohjelmoinnin opetuksen myönnetään olevan vaikea tehtävä sen ensimmäisestä toteutuksesta lähtien. Oppilaat näyttäisivät pitävän ohjelmointia tylsänä ja väsyttävänä toimintana. Tämä ongelma saattaisi ratketa, jos ohjelmoinnin aloituskurssi tehtäisiin helpoksi ja viihdyttäväksi oppilaille. (Papadakis ym., 2015)

Ohjelmointia voidaan suorittaa erilaisia ohjelmointikieliä ja ohjelmointiympäristöjä käyttäen. Ohjelmointikielet voidaan karkeasti jakaa visuaalisiin ja tekstipohjaisiin kieliin. On tehty paljon tutkimuksia ja kokeiluja, joissa on pyritty luomaan yläkouluikäisiä oppilaita kiinnostavia tehtäviä ja ympäristöjä, joiden avulla ohjelmoinnin perusteita voidaan opetella.

Tässä luvussa esitellään ja vertaillaan ensiksi Suomessa yleisimmin yläkoulujen ohjelmoinnin alkeisopetuksessa käytettäviä ohjelmointikieliä. Tämän jälkeen esitellään muita ohjelmoinnin alkeisopetukseen sopivia ympäristöjä.

5.1 Visuaaliset ohjelmointikielet

Visuaalisilla ohjelmointikielillä ohjelmointi tapahtuu graafisessa ympäristössä erilaisia moduleita yhdistämällä. Toisin kuin tekstipohjaisia ohjelmointikieliä käytettäessä ohjelmoijan ei tarvitse kirjoittaa ohjelmointilausekkeita. Tämän johdosta nämä kielet sopivat hyvin ensimmäisiksi ohjelmointikieliksi ja niitä käytetään jo ala-asteen ensimmäisistä luokista alkaen. Yleisin suomalaisissa kouluissa käytettävä visuaalinen ohjelmointikieli on Scratch.

Bau ym. (2017) näkevät, että käyttämällä koodauslohkoja ohjelmoinnin opetukseen seuraavat koodaamisen oppimisen esteet pystytään hallitsemaan paremmin:

- ohjelmoinnin sanaston oppiminen on vaikeaa. Lohkot helpottavat tätä ohjelmaa, koska lohkon valitseminen paletista on helpompaa kuin sanojen muistaminen.

- koodia on vaikea käyttää, koska se tuo korkean älyllisen (kognitiivisen) kuorman aloitteleville ohjelmoijille. Lohkot vähentävät älyllistä kuormaa kasaamalla koodia vähempään määrään merkityksellisiä elementtejä.
- koodaus on altista virheille. Lohkot auttavat välttämään tiettyjä perusvirheitä pakottamalla koodauksen tiettyyn rakenteeseen.

Visuaalisten ohjelmointikielten etuja on mm. se, ettei ohjelmoijan tarvitse tuntea ohjelmointikielen syntaksia eikä ohjelman käännösvirheitä (compile errors) esiinny. Toisaalta visuaalisilla ohjelmointikielillä on myös rajoituksia, kuten että ne eivät anna mahdollisuutta kehittyneemmille oppilaille taitojen edelleen kehittämiselle eivätkä ne anna oppilaille käsitystä ohjelman kehitysprosessista alhaalta ylöspäin (ohjelmakomentojen valinta ja niiden yhdistäminen). (Tsukamoto ym., 2015)

5.1.1 Scratch

Scratch on visuaalinen ohjelmointikieli ja -ympäristö, jonka avulla voidaan luoda mediaa ja koodia sisältäviä projekteja. Ohjelmointi tapahtuu yhdistämällä erilaisia komentolohkoja. Scratchin avulla käyttäjät voivat luoda vuorovaikutteisia ja ääntä, kuvaa tai videoita sisältäviä ja yhdistäviä ohjelmointiprojekteja, kuten animaatioita, pelejä, onnittelukortteja, musiikkivideoita, tiedeprojekteja ja simulaatioita (Maloney ym., 2010). Scratch on käännetty 40 kielelle ja se on käytössä yli 150 maassa (Scratch, 2019), Suomessakin Scratch on suosittu kieli ohjelmoinnin opetuksessa (Marttala, 2017).

Kalelioglu ja Gülbahar (2014) toteavat tutkimuksessaan, että on lukuisia oivalluksia ja toiveita herättäviä löydöksiä Scratchin käyttämisestä ohjelmointitaitojen ja -konseptien opettamiseen. He uskovat, että Scratchin käyttö jatkuu 8–18 vuoden ikäisten oppilaiden opetuksessa tuoden jatkossakin positiivisia tuloksia.

5.2 Tekstipohjaiset ohjelmointikielet

Tekstipohjaisilla ohjelmointikielillä ohjelmointi tapahtuu ohjelmointilausekkeita kirjoittamalla. Kielten syntaksi voi olla hankalaa ja niiden käyttö esimerkiksi ala-asteella tapahtuvaan ohjelmoinnin opetukseen voi olla hyvinkin haastavaa. Toisaalta tekstipohjaiset ohjelmointikielet ovat joustavampia kuin visuaaliset ja mahdollistavat erilaisten ohjelmien kirjoittamisen. Ne vaativat ohjelmoijalta enemmän laskennallista ajattelua. Suomalaisissa kouluissa yleisimmin käytetyt tekstipohjaiset ohjelmointikielet ovat Python ja Javascript.

5.2.1 Python

Python on tulkittu, vuorovaikutteinen olio-ohjelmointikieli (Sanner, 1999). Mannila ja de Raadt (2006) ovat tehneet tutkimuksen, jossa he vertasivat 11 eri ohjelmointikielen soveltuvuutta ohjelmoinnin aloituskurssille. Tutkimuksessa ohjelmointikielet on vertailtu 17 eri arviointiperusteen mukaan. Nämä on jaettu neljään eri pääalueeseen:

- oppiminen sisältää perusteet, jotka liittyvät ohjelmointikielen käyttämiseen opetuksessa
- suunnittelu ja ympäristö sisältää perusteet, jotka liittyvät kielen suunnitteluun sekä ympäristöön, jossa kieltä käytetään
- tuki ja saatavuus kuvaa tukiyhteisön ja kielen saatavuuden sekä resurssit kielen opetukseen
- alkeiskurssin jälkeen -perusteet kuvaavat kielen käyttämistä myöhemmissä, eri tasoisissa ohjelmointikursseissa

Mannilan ja de Raadt’n (2006) johtopäätös vertailun tulosten pohjalta on, että parhaiten ohjelmoinnin alkeisopetukseen sopivat Python ja Eiffel. Heidän mukaansa tämä ei ole yllättävää, koska nämä kielet on suunniteltu pitäen opetus mielessä. Tutkimuksen mukaan myös kaupalliseen tarkoitukseen suunnitellulla Java-kielellä on omat hyvät puolensa opetuskielenä.

Mészárosová (2015) on tutkinut Pythonin soveltuvuutta ohjelmoinnin alkeiskurssin opetukseen ja verrannut sitä monessa paikassa käytössä olevaan Pascal-kieleen. Hänen mukaansa ohjelmoinnin opetuksen painopisteen tulisi olla algoritmien opettamisessa ja ongelmanratkaisukykyjen kehittämisessä eikä ohjelmointikielen opetuksessa, vaan pikemminkin ohjelmointitaitojen perusteiden opetuksessa. Tähän tarkoitukseen sopivan ohjelmointikielen tulee antaa oppilaiden ja opettajien keskittyä algoritmeihin ja ohjelmointitaitojen hankkimiseen ilman, että heidän täytyy kuormittaa itseään kielen syntaksilla ja monimutkaisella kehitysympäristöllä. Opettajien vaatimukset ensimmäiselle ohjelmointikielelle tulivat selviksi ja vaikka Python ei täyttänyt aivan kaikkia vaatimuksia, katsottiin sen kuitenkin olevan muita vertailukieliä parempi vaihtoehto.

Mészárován (2015) mukaan Englannissa, Tsekissä ja Unkarissa annetaan yläasteella ja lukiossa (secondary school) ohjelmoinnin alkeisopetusta ja osa kouluista on siirtynyt käyttämään Python-kieltä tähän opetukseen.

5.2.2 JavaScript

JavaScript-ohjelmointikieltä käytetään ennen kaikkea lisäämään interaktiivisuutta websivuilla, kuten hiiren klikkauksia tai syöttötietojen kirjoittamista. JavaScript on dynaaminen komentokieli (script language), mikä tarkoittaa sitä, että se tulkataan suoraan sen sijaan, että ohjelma käännettäisiin ennen sen suorittamista. (Korpela, 2009)

JavaScriptin yksinkertaisuus, luonnolliset rajapinnat ja saumaton integrointi web-sivuihin tekevät aloitteleville ohjelmoijille mahdolliseksi tehdä mielenkiintoisia ohjelmia nopeasti (Reed, 2001).

De Raadt (2008) mainitsee mm. seuraavat edut JavaScriptin käytölle ohjelmoinnin alkeisopetuksessa:

- mahdollisuus oppia tärkeät käsitteet nopeasti, koska JavaScriptissä on helppokäyttöiset palvelut käyttäjän tiedon syöttämiseen ja tulostamiseen sekä tietojen tallennukseen.

- yksinkertainen käyttää, koska web-selaimet tulkkaavat JavaScriptiä osana web-sivua eikä ohjelman kääntämistä tarvita.
- houkutteleva käyttäjille, koska JavaScriptiä käytetään dynaamisten web-sivujen tekemiseen.

5.2.3 Processing

e-Oppi-oppimateriaaleissa käytetty, Java-kieleen perustuva Processing on erityisesti kuvien generoimiseen ja muokkaamiseen suunniteltu ohjelmointikieli, jolla on helppo ohjelmoida esimerkiksi kuvia ja animaatioita (Reas & Fry, 2006). Tsukamoto ym. (2015) pitävät Processingia sopivana kielenä ohjelmoinnin alkeisopetukseen, koska Processing-ohjelmointi 1) ei vaadi luokkien määrittämistä, 2) ei vaadi koodia luomaan ikkunoita visuaaliseen ohjelmointiin, 3) tarjoaa monia valmiita lausekkeita erilaisten kuvioiden piirtämiseen helpolla tavalla, 4) tarjoaa keinon tarkistaa varattujen sanojen oikeinkirjoitus, ja 5) tarjoaa yksinkertaisen tavan tehdä animaatioita.

5.2.4 Racket

Racket on funktionaalinen ohjelmointikieli. Se luotiin ohjelmoinnin opetuskieleksi, joka soveltuu hyvin aloittelijoiden ohjaamiseen. Lisäksi Racket-kielen kehittäjät halusivat luoda pedagogisesti hyvän ohjelmointiympäristön, joka sopii nimenomaan aloittelijoille eikä ole ainoastaan ammattilaisille tarkoitettu työkalu. (Felleisen ym., 2015)

5.3 Ohjelmoinnin opetusympäristöt

5.3.1 Code.org

Vuonna 2013 kaksosveljesten Hadi ja Ali Paretovin perustama Code.org on voittoa tavoittelematon organisaatio, jonka tarkoituksena on tarjota oppimateriaaleja tietojenkäsittelyn opetukseen. Code.org sisältää laajan valikoiman erilaisia ilmaisia kursseja 4–18 vuotiaille oppilaille ja osa näistä kursseista (lähinnä alakouluun suunnatut) on saatavilla suomenkielisinä. Kurssit on toteutettu web-applikaatioina ja niiden käyttöön tarvitsee selaimen ja nettiliittymän. Organisaatiolla on yli 100 yhteistyökumppania ja sen kursseja

käyttävät kymmenet miljoonat oppilaat ja miljoona opettajaa ympäri maailman (Code.org, 2020). Code.org-sivusto tukee sitä, että oppilaat voivat suorittaa kursseja itsenäisesti tai vaikka pariohjelmointina. Opettajan on mahdollista seurata reaaliajassa oppilaiden edistymistä.

5.3.2 Codecademy

Codecademy on interaktiivinen ja yhteisöllinen ohjelmoinnin opetukseen ja oppimiseen käytettävä alusta (Ma, 2013). Codecademy tarjoaa ilmaisia verkkokursseja monista eri ohjelmointiaiheista ja sillä on noin 25 miljoonaa käyttäjää (Sharp, 2019).

Sharp (2019) on tutkinut Codecademyn käyttöä Python-ohjelmoinnin alkeiskurssin osana. Hänen kokemuksensa näiden materiaalien käytöstä oppikirjojen lisänä olivat positiivisia. Käydyt kurssit tarjosivat riittävän tiedon Python-kielen syntaksista. Lisäksi kurssit mahdollistivat oppilaille koodauksen harjoittelua, johon järjestelmä antoi välitöntä palautetta. Tämän lisäksi hän nosti positiivisina asioina esiin oppilaiden mahdollisuuden käytä kursseja omaan tahtiinsa sekä sen, ettei kurssien käyminen maksanut mitään. Codecademyn kurssien negatiivisena puolena Sharp (2019) näki sen, että kurssit tarjosivat hyvin rajoitetun mahdollisuuden luovuuden käyttöön, koska tehtävien oikeat vastaukset ovat ennalta määriteltyjä. Toisena negatiivisena asiana hän mainitsi oppilaiden tekevän kursseja yritä-ja-erehdy -periaatteella sen sijaan, että olisivat käyttäneen ajattelun taitojaan.

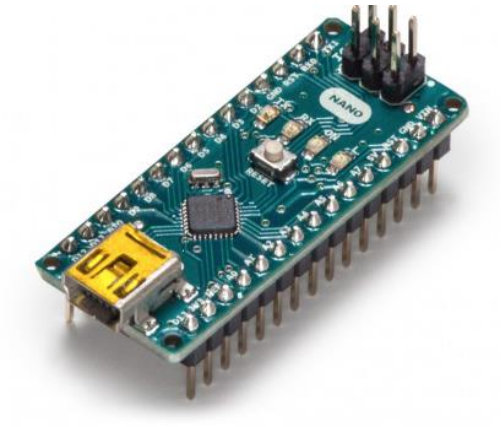
Codecademyn kursseja ei ole saatavilla suomenkielisinä, mikä saattaa rajoittaa niiden käyttöä.

5.4 Elektroniikka-alustat

5.4.1 Arduino

Arduino on avoimeen lähdekoodiin, helppokäyttöiseen ohjelmistoon ja laitteistoon perustuva elektroniikka-alusta, jolla voi lukea syötteen (input) – sensorin valon, napin pai-

nalluksen tai Twitter-viestin – ja muuttaa se ulostuloksi (output), joka voi olla esimerkiksi moottorin käynnistäminen, led-valon sytyttäminen tai jokin julkaisu verkossa (Arduino, 2016). Arduino sisältää ominaisuuksia, jotka ovat tehneet siitä ideaalisen opetustarkoituksiin: 1) se on suhteellisen halpa (mallista riippuen noin 25–35 €), 2) sitä on helppo käyttää ja ohjelmoida, 3) se sisältää avoimen ja laajennettavan ohjelmisto- ja laitteistoalustan ja 4) sille on saatavissa laaja määrä projekteja ja kirjastoja (Orfanakis & Papadakis, 2016). Kuva 9 näyttää Arduino-elektroniikka-alustan rakenteen. Arduino koostuu yhdestä piirikortista, jolla on erilaisia sisäänmeno- ja ulostuloliitäntöjä.



Kuva 9. Arduino Nano -elektroniikka-alusta. Lähde: store.arduino.cc/arduino-nano.

Orfanakis ja Papadakis (2016) esittelevät harjoitustyön, jonka tekeminen perustuu Twitterin, Python-ohjelmointikielen, Arduino-mikrokontrollerin ja kahvinkeittimen käyttöön. Twitter on yhteisö ja mikroblogipalvelu, jonka käyttäjät pystyvät lähettämään ja lukemaan toistensa korkeintaan 280 merkkiä pitkiä tekstipohjaisia viestejä internetissä (Isotalus ym., 2018). Orfanakisin ja Papadakisin (2016) projektissa oppilaat tekivät 3−4 hengen ryhmissä Twitter-tilin, jolta lähetettiin viesti ‘Start’ tietokoneelle, jolle ohjelmoitiin Python-kielellä ohjelma, joka lähetti USB-portin kautta viestin Arduino-alustalle. Arduino puolestaan ohjasi relettä, joka käynnisti kahvinkeittimen. Vastaavasti lähettämällä viesti ‘Stop’ kahvinkeitin sammutettiin.

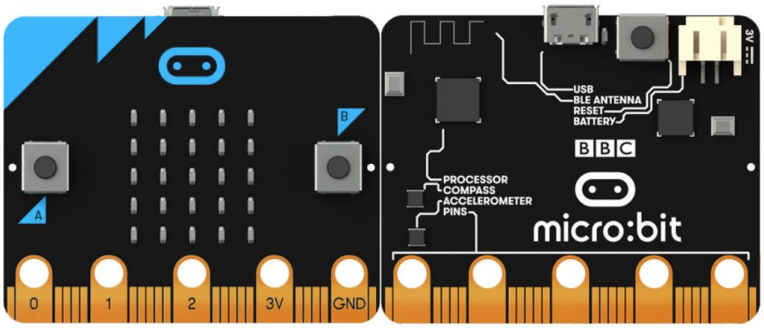
Projektin tarkoituksena oli motivoida oppilaat käyttämällä helppoa ja oppilaiden mielenkiintoa houkuttelevaa sisältöä. Harjoitustehtävä antaa oppilaille kosketuksen julkisuudessa esillä oleviin esineiden Internet (Internet of Things) ja älykäs koti -teknologioihin. Tarvittavan laitteiston, Arduino-alustan ja releen, kustannukset olivat noin 25 euroa.

5.4.2 Micro:bit

Arduinon lisäksi toinen yleisesti opetuksessa käytettävä elektroniikka-alusta on Iso-Britanniassa kehitetty Micro:bit-pienoistietokone, jota käytetään kouluissa mm. Suomessa, Islannissa, Singaporessa ja Sri Lankassa. Micro:bitin ohjelmointi tapahtuu web-selaimen avulla ja ohjelmointikielenä voidaan käyttää Pythonia, JavaScriptiä tai graafista ohjelmointia (koodilohkoja). Laitteeseen löytyy yli 200 valmista tehtävää tai harjoitusta (Microbit, 2019). Helsingin yliopiston kasvatustieteellisen tiedekunnan koordinoiman Innokas-verkoston (www.innokas.fi) sivuilta löytyy paljon erilaisia materiaalipaketteja Micro:bitin käyttöön opetuksessa.

Sentance ym. (2017) ovat tutkineet Micro:bitin käyttöä kouluissa ja listaavat seuraavat hyödyt tällaisen todellisen tietokonelaitteen käyttämisestä ohjelmoinnin opetuksessa:

- Motivaatio. Motivaatio lisääntyy, koska oppikokemus ja -tulos ovat konkreettisia ja näkyviä, eivät virtuaalisia.
- Käsinkosketeltavuus. Fyysisten laitteiden konkreettisuus auttaa opiskelijoita luomaan luonnollisia yhteyksiä. Iteratiivisesti tehtävät järjestelmien virheenkorjaus ja parantaminen auttavat ymmärtämään ohjelmointikäsitteitä ja ohjelmoinnin kehitysprosessia.
- Yhteistyö. Laitteiden kanssa työskentely soveltuu hyvin ryhmätyöhön, koska eri roolit voivat sisältää esimerkiksi käyttötapausten, laitteistorajapintojen, algoritmien ja käyttöliittymän suunnittelua.
- Luovuus. Oppilaat suhtautuvat luonnollisesti tehtävän konkreettiseen luonteeseen vapauttaen luovuuden siihen, mitä he rakentavat.



Kuva 10. Micro:bit pienoistietokone. Lähde: www.microbit.org.

5.5 Robotiikka

Robottien käyttö opetuksessa on yleistä, sillä niiden nähdään tuovan uusia hyötyjä luonnontieteiden, teknologian ja matematiikan opetukseen. Robotit ovat tehokas keino motivoimaan ja tukemaan opetusta. Lisäksi niiden käytön on havaittu parantavan oppilaiden sosiaalisia kykyjä ja yhteistyötaitoja. (Johnson, 2003) Benitti (2012) on kuitenkin havainnut kirjallisuustutkimuksessaan, että suurin osa robottiteknologian käytöstä opetuksessa keskittyy tukemaan aineita, jotka on läheisesti kytketty robotiikkaan, kuten robottien ohjelmointi, rakentaminen ja mekatroniikka osana matematiikan ja fysiikan opetusta. Lisäksi useimmissa tapauksissa robotteja on käytetty passiivisina työkaluina, joita on vain rakennettu ja ohjelmoitu.

Oddie ym. (2010) mukaan robotiikan käyttö ohjelmoinnin opetuksessa on hyödyllistä monin tavoin, esimerkiksi:

- Sitoo kaikki oppilaat keskeisten tehtävien tekemiseen tarjoten samalla lisämahdollisuuksia edistyneemmille oppilaille.
- Auttaa lieventämään ohjelmoinnin abstraktia luonnetta tarjoamalla konkreettisen ympäristön palautteen näkemiseksi.
- Rohkaisee kokeilua ja tutkimista, esimerkkinä anturien käyttö.
- Tarjoaa visualisointeja ongelmien ratkaisuista.
- Tarjoaa ympäristön ohjelmointikonseptien oppimiseen ja opettaa luomaan vikatilanteita kestävää koodia, esimerkiksi oppilaat eivät halua, että robotti putoaisi työpöydältä tai törmäisi seinään.

- Ohjelmoinnin sanaston oppiminen mielenkiintoisella tavalla.

Rusk ym. (2008) esittävät uusia strategioita siitä, miten robottiteknologiaa voitaisiin esitellä oppilaille. Heidän mukaansa onnistuneita strategioita ovat olleet: 1) keskittyminen teemoihin, 2) tekniikan ja taiteen yhdistäminen, 3) tarinoiden kertominen ja 4) näyttelyjen järjestäminen kilpailujen sijaan. Monet oppilaat, jotka eivät ole olleet kiinnostuneita perinteisiin robottien käyttötapoihin, motivoituvat paremmin, kun heidän tulee kertoa tarina (esimerkiksi nukketeatteri robottien esittämänä) tai käyttää robotteja yhdistettynä musiikkiin tai kuvataiteeseen.

5.5.1 Lego Mindstorms

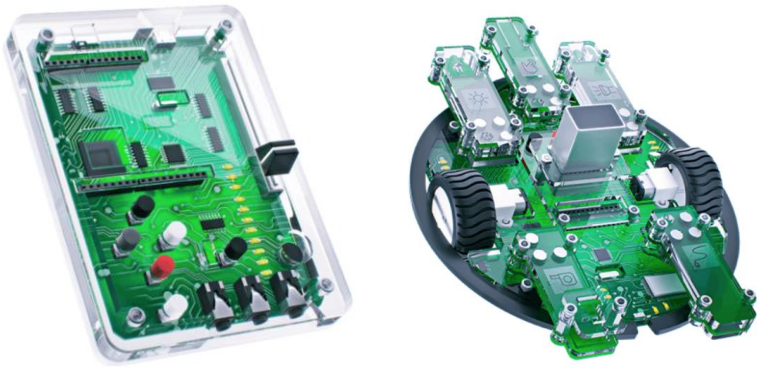
Lego Mindstorms on Legon koulutuskäyttöön kehittämä robotti, jota ohjelmoidaan Legon graafisella EV3-ohjelmointikielellä. Lego-robotteja on kehitetty jo useita vuosikymmeniä, joiden aikana niihin on tuotu ominaisuuksia, joiden ansiosta ne soveltuvat hyvin opetuskäyttöön esikoulusta yliopistotasolle (Pihola, 2016). Lego-roboteista on julkaistu useita versioita, joista viimeisin on Lego Mindstorms EV3. Lego-robotteihin on saatavilla paljon tukimateriaalia ja erilaisia harjoitustehtäviä liittyen esimerkiksi liikkeen ohjaamiseen ja erilaisten antureiden käyttöön. Garcia-Peñalvo ym. (2016) listaavat Lego-robottien hyvinä ominaisuuksina joustavuuden, ohjelmoinnin helppouden, tehokkuuden, suosion ja laajan määrän käyttömahdollisuuksia esikoulusta yliopistotasolle. Haittoina he mainitsevat korkean hinnan ja EV3:n tiettyjen osien helpon rikkoutumisen.

Oppilaiden motivaatiota robottien ohjelmoimiseen voi lisätä osallistuminen erilaisiin kilpailuihin. Erilaisia Lego-robottien ohjelmointikilpailuja järjestetään säännöllisesti oppilaille, esimerkkinä FIRST LEGO League (FLL), johon osallistuu maailmanlaajuisesti vuosittain noin 300 000 nuorta (Leino, 2015).

Kuva 11. Lego Mindstorms EV3 -robotti. Lähde: education.lego.com/en-us/products/lego-mindstorms-education-ev3-core-set/5003400

5.5.2 Robbo

Robbo-robottia tarjotaan kahtena erilaisena tuotteena, Lab ja Robot Kit. Lab on Arduino-alustalle rakennettu robotti, joka ei pysty liikkumaan itsekseen. Sen avulla pystytään kuitenkin tutustumaan erilaisiin ilmiöihin, kuten valoon ja ääneen. Esimerkiksi Labin avulla voidaan rakentaa liikennevalot tai tutustua siihen, miten tietokoneen näppäimistö on ohjelmoitu. Lab sisältää mm. mikrofonin, valosensorin, kaiuttimen, 11 ledvaloa ja lineaarisesti säädettävän vastuksen. Sitä voidaan ohjelmoida Arduinon ohjelmointirajapinnan kautta. Robot Kit on edistyneempi kahdella moottorilla ja useilla erilaisilla sensoreilla vastustettu robotti. Myös Robot Kit on rakennettu Arduino-alustalle (Robbo, 2019). Kuvassa X on näytetty Robbo-robottien rakenteet.



Kuva 12. Robbo Lab (vasemmalla) ja Robbo Robot Kit. Lähde: www.robbo.world

Garcia-Peñalvo ym. (2016) listaavat Lab-robottien vahvuuksina suhteellisen edullisen hinnan, kestävyyden ja mahdollisuuden ohjelmoida sekä Scratchin että Arduinon avulla. Heikkouksina on rajoitettu joustavuus. Robot Kitin vahvuuksina mainitaan puolestaan kestävyys, useat erilaiset sensorit ja mahdollisuus ohjelmoida sekä Scratchin että Arduinon avulla. Heikkouksina ovat puolestaan kallis hinta ja rajoitettu joustavuus.

5.5.3 Muut robotit

Opetuskäyttöön on suunniteltu lukuisa määrä muitakin robottituotteita. Jotkut näistä soveltuvat hyvin ohjelmoinnin alkeisopetukseen, mutta yläkoulun oppilaat voivat pitää niiden ohjelmoimista hieman lapsellisena. Tällaisia robotteja ovat esimerkiksi ohjelmoitava pallo Sphero, värikäs lapsille suunniteltu Bee-Bot sekä värien avulla ohjattava Ozobot.

Kehittyneempiä Lego Mindstormin tapaisia robotteja ovat esimerkiksi lohkoista koottava Cubelets, edullinen moottorin ja sensoreita sisältävä Edison, jota voidaan laajentaa legopalikoilla sekä Arduino-alustalle rakennettu mBot.

5.6 Älypuhelimen ohjelmointi

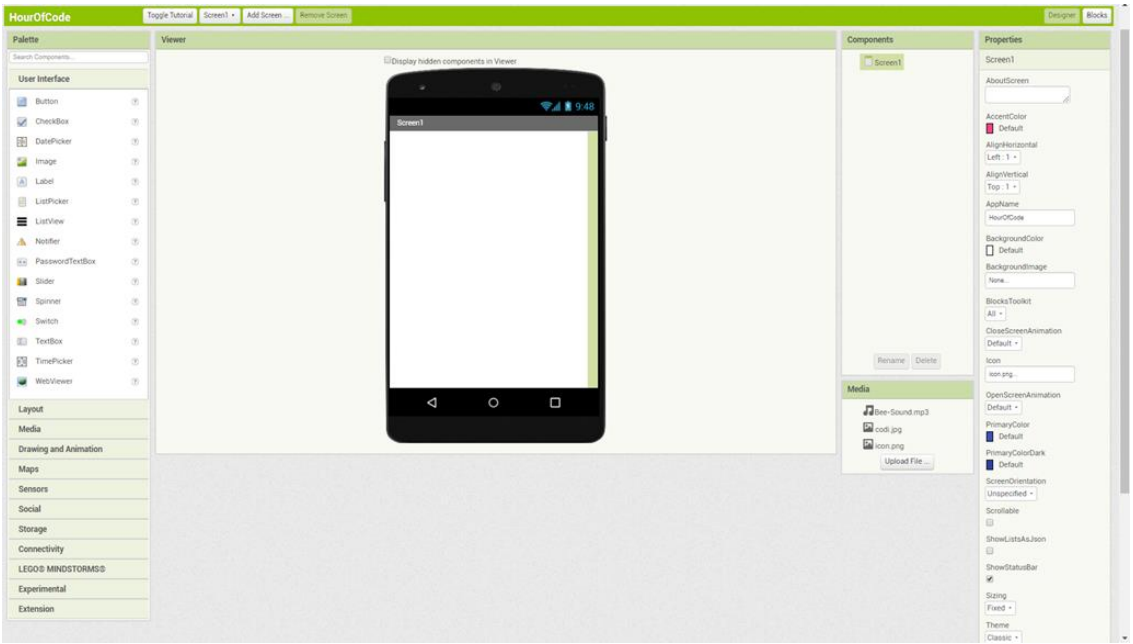
Oppilaat oppivat parhaiten, kun heidän oppimiskohteensa on kytketty asioihin, joita he tekevät päivittäin. Useimmat teini-ikäiset käyttävät säännöllisesti älypuhelimia, jotka tarjoavatkin näin ollen ainutlaatuisen mahdollisuuden heidän sitouttamiseensa. (Wagner ym., 2013)

Googlen kehittämä App Inventor for Android (AI) on visuaalinen ohjelmointiympäristö, jossa voidaan toteuttaa ohjelmia Android-älypuhelimiin. AI sisältää lohkoja ja sen ohjelmointi toimii vedä-pudota (drag-drop) -periaatteella. Siihen kuuluu myös emulaattori, jonka avulla on mahdollista ajaa tehtyjä ohjelmia tietokoneella. Tehtyjä ohjelmia voidaan myös testata joko Android-puhelimella tai -tabletilla (Wagner ym., 2013). Näissä ohjelmissa voidaan sekoittaa toisiinsa sosiaalista verkottumista, paikannustietoa ja web-pohjaista tiedonhakua (Abelson, 2009).

App Inventorin käyttämisestä aloittelijoiden ohjelmoinnin opetukseen on tehty monia tutkimuksia. Näissä on käytetty vaikeusasteeltaan erilaisia ja erityyppisiä harjoitustehtäviä. Esimerkiksi Hsu ym. (2012) ovat käyttäneet opetustapaa, jossa tutustutaan olemassa oleviin ohjelmiin, joihin oppilaat tekevät muutoksia. Seuraavia ohjelmia on käytetty tähän tarkoitukseen:

1. HelloPurr on yksinkertainen sovellus, jonka avulla oppilaat tutustuvat App Inventoriin. Sovellus toteuttaa kissan äänen nappia painettaessa.
2. PaintPot mahdollistaa eriväristen viivojen piirtämisen näytölle sormella. Oppilaat ohjelmoivat sovellusta muistamaan piirretyn värin.
3. MoleMash on peli, jossa pelaajat koskettavat satunnaisesti esiin tulevia myyriä kerätäkseen pisteitä. Tämän sovelluksen avulla opetetaan ohjelmoimaan kuvia ja satunnaismuuttujia.
4. NoTextingWhileDriving-ohjelmassa käyttäjät vastaavat tekstiviesteihin lähettäen ennalta määriteltyjä viestejä koskettamatta puhelintaan. Käyttäjät tutustuvat ohjelmoimaan viestejä ja tietokantakomponentteja.
5. ParisMapTour-ohjelmassa käyttäjät valitsevat listalta paikkojen nimiä ja linkittävät nämä Google Maps -ohjelmistoon nähdäkseen kuvia kaduilta. Presidents Quiz -pelissä käyttäjille esitetään tietokilpakysymyksiä USA:n presidenteistä. Näiden molempien ohjelmien avulla opetetaan tietolistojen käyttöä käyttäjän valinnan mukaan.

App Inventorin käytöstä tehdyissä tutkimuksissa oppilaat vaikuttavat olleen hyvin motivoituneita oppimaan ohjelmointia tekemällä ohjelmia, joita he voivat käyttää omissa puhelimissaan. Wolber (2012) näkee App Inventorin yhdistävän alhaisen kynnyksen ohjelmoinnin aloittamiseen kykyyn luoda sovelluksia, joilla on todellista arvoa. Hänen mukaansa ohjelmointikielen visuaalisuus vähentää aloittelijoiden ongelmia syntaksin kanssa ohittaen näin ensimmäisen esteen ohjelmoinnin oppimiselle. Oppilaat voivat melkein heti rakentaa todellisia, hyödyllisiä sovelluksia, mikä motivoi heidät kohtaamaan ohjelmoinnin oppimisen toisen esteen: oppia ratkaisemaan loogisia ongelmia ja määrittämään interaktiivista käyttäytymistä. Helsingin yliopiston tietojenkäsittelytieteen osastolla on kehitetty App Inventorin harjoitustehtäviä 5–6 ja 7–9 luokka-asteille. Nämä tehtävät sisältävät oppimateriaalia niin opettajille kuin oppilaillekin. Harjoitustehtävinä löytyy erilaisia yksinkertaisia pelejä, kuten kallistelupeli ja lentelypeli sekä jotain ohjelmia hyötykäyttöön, kuten ostoslista ja kompassi.



Kuva 13. AppInventor-ohjelmointiympäristö. Lähde: ai2.appinventor.mit.edu

6 Ohjelmoinnin opetuksen didaktiikka

Tässä luvussa esitetään ohjelmoinnin opetuksen didaktiikasta tehdyn kirjallisuustutkimuksen tuloksia. Tutkimus on jaettu kolmeen osaan, ensimmäiseksi esitellään ohjelmoinnin opetuksen pohjana oleva teoreettinen malli, konstruktivismi. Tämän jälkeen esitellään ohjelmoinnin opetukseen liittyviä tutkimuksia ja kolmantena tutkimusalueena on opetussuunnitelmassakin mainittu ajattelumallien opettaminen.

6.1 Konstruktivismi

Tietotekniikan opetus ja oppiminen pohjautuu yleisesti kahteen olennaisesti erilaiseen teoreettiseen malliin, objektivismiin ja kontruktivismiin. Objektivismi, jota kutsutaan myös opettajakeskeiseksi opetukseksi, uskoo oppimisen tapahtuvan, kun oppilaat kuuntelevat opettajan opetusta, osallistuvat täydentäviin harjoituksiin ja vastaavat ulkoiseen motivointiin. Konstruktivismi puolestaan keskittyy enemmän oppilaan oppimiseen ja kokemuksiin ja oppiminen vaatii oppilaan aktiivisesti rakentavan oman tietämyksensä. (van Gorp ja Grissom, 2001)

Konstruktivismi on oppilaskeskeinen pedagogia, jolle on tyypillistä oppiminen kokeilemisen kautta (Wulf, 2005). Tämän opetustavan yleisti Seymour Pappert, joka ehdotti, että matematiikan ja luonnontieteiden opetus olisi tehokkaampaa, jos oppilaille tarjottaisiin rikas, kognitiivinen oppimisympäristö tutkimusta varten. Tutkimisen aikana oppilaat loisivat oman tietämyksensä asioista, joihin he kytkevät oppimisensa. Konstruktivistinen oppiminen on usein luonteeltaan yhteisöllistä oppilaiden työskennellessä yhdessä osana tietämyksen hankkimisprosessia (Wulf, 2005). Konstruktivismin teorian mukaan oppilaat rakentavat (construct) osaamistaan pikemminkin kuin saavat ja tallentavat opettajan välittämää tietoa (Ben-Ari, 1998).

Ohjelmoinnin opetus sisältää monia konstruktivistiseen ja yhteisölliseen oppimiseen pohjautuvia toimintoja, kuten koodin läpikäynti, koodin kirjoittaminen, koodin lukeminen, virheiden etsiminen koodista ja tuettu ongelmien ratkaisu (van Gorp ja Grissom,

2001). Hadjerrouitin (2008) tutkimuksen mukaan pätevyys ohjelmoinnissa vaatii korkeamman tason ajattelun taitoja, kuten analysointia, suunnittelua, analyyttistä ajattelua, uudelleenkäyttöä, arviointia ja reflektiota. Van Gorpin ja Grissomin (2001) mukaan konstruktivismi auttaa oppilaita oppimaan ohjelmointia, koska oppiminen perustuu ongelmanratkaisuun.

Wulf (2005) on kehittänyt konstruktivistiseen pedagogiaan pohjautuvan mallin ohjelmoinnin opettamiseen. Hänen oppituntinsa koostuu seuraavista vaiheista:

1. Alustava tutustuminen. Oppilaiden alustava tutustuminen materiaaliin tapahtuu yleensä lukemalla yksin. Tätä edeltää opettajan lyhyt esittely luettavasta aiheesta.
2. Lyhyt kertaus. Alustavaa tutustumista seuraa lyhyt kertaus, joka tehdään opettajan johdolla esimerkiksi kysymysten ja vastausten kautta.
3. Opastettu harjoittelu. Oppilaille annetaan tehtäväksi konkreettinen harjoitustehtävä. Oppilaat voivat työskennellä tässä vaiheessa niin yksittäin, pareittain tai ryhmissä.
4. Yksilöllinen tai ryhmässä tehtävä ohjelmointityö. Oppilaat osoittavat opetetun asian hallinnan soveltamalla sitä koodia kirjoittamalla. Tarvittavaa tukea annetaan ja tehtävä voidaan suorittaa esimerkiksi siten, että osa koodista on annettu etukäteen.
5. Oppimissaavutuksen arviointi. Tätä vaihetta ei tehdä erillisenä jokaisen opetetun aiheen jälkeen. Monissa tapauksissa tehdyn harjoitustyön tulos toimii arvioinnin perusteena.

Crawfordin (1999) mukaan kontruktivismi on ainoa toimiva menetelmä tietotekniikan opettamiseen englantilaisissa toisen asteen kouluissa, koska opettajilla ei voi olla kaikkea opetettavaan aiheeseen liittyvää sisältöä. Lisäksi teknologia muuttuu hänen mukaansa niin nopeasti, että oppilailla voi olla joistakin osa-alueista parempi tieto kuin opettajalla.

6.2 Ohjelmoinnin opetus

Ohjelmoinnin opetusta on tutkittu omana aiheenaan jo kymmeniä vuosia. Nämä tutkimukset ovat keskittyneet ennen kaikkea ohjelmointikielten ja -ympäristöjen käyttöön, jonkin verran myös erilaisiin opiskelutapoihin. Suomessa pääosa tutkimuksista on kohdistettu joko alakoulujen tai yliopistojen opetukseen – yläkoulujen ohjelmoinnin opetus vaikuttaa olevan vähemmän tutkittu aihealue. Harvat tutkimukset ovat keskittyneet ohjelmoinnin opetuksessa käytettäviin opetusmateriaaleihin, mutta ne ovat välillisesti mukana useissa tutkimuksissa. Joitakin tutkimuksia on jo ehditty tehdä myös oppimateriaaleista, kuten Uotilan (2016) pro gradu -tutkielma ohjelmoinnin sisällyttämisestä peruskoulun matematiikan oppikirjoihin.

Ohjelmointia on opetettu jo useita vuosia osana perusopetusta esimerkiksi Japanissa, Etelä-Koreassa, Uusi-Seelannissa, Iso-Britanniassa, Yhdysvalloissa, Israelissa ja Virossa (Tsukamoto ym., 2015), joten aiheesta on myös tehty monia tutkimuksia. Pears ym. (2007) ovat puolestaan listanneet yli 180 tutkimusta kirjallisuustutkimukseen, jossa he pyrkivät auttamaan aloittelijoiden ohjelmointikurssien suunnittelijoita tutkimalla mm. opetuksessa käytettyjä ohjelmointikieliä, työkaluja, ympäristöjä ja pedagogioita. Waite (2017) puolestaan kertoo tutkineensa yli 700 eri lähdettä koulujen tietotekniikan opetuksessa käytettävää pedagogiikkaa koskevassa tutkimuksessaan.

Olen tutkinut ohjelmoinnin opetusta lyhyesti myös ‘Opettaja työnsä tutkijana’ -tutkielmassa (Haapakangas, 2019), johon on sisällytetty seuraavaksi esiteltävät Sentancen & Csizmadian (2015) ja Saeli ym. (2010) tutkimukset.

Sentance & Csizmadia (2015) tekemän tutkimuksen perusteella opettajien käyttämät ohjelmoinnin opetuksen strategiat jaettiin viiteen eri kategoriaan:

1. Kontekstisoitu oppiminen, jossa tietojenkäsittelyn opiskelu liitettiin esimerkiksi jonkin muun aineen opiskeluun.

2. Laskennallisen ajattelun kehittäminen. Tutkimuksessa opettajat halusivat tuoda esille monia käyttämiään laskennallisen ajattelun konsepteja ja prosesseja, esimerkkeinä looginen (algoritminen) ajattelu, abstraktio, erittely (decomposition) ja ongelmanratkaisu.
3. Koodin muokkaus. Monia eri keinoja käytettiin ohjelmakoodin ymmärtämisen oppimiseen, kuten oppilaille annettiin valmista koodia edelleen laajennettavaksi sekä vikojen etsimistä varten.
4. Yhteistyössä työskenteleminen, joka sisälsi useita eri toimintatapoja, kuten tiimityöskentely, mentorointi ja pariohjelmointi.
5. Oppiminen ilman tietokonetta. Merkittävä osa opettajista mainitsi käyttävänsä ilman tietokonetta tehtäviä visualisointikeinoja.

Saeli ym. (2010) mukaan ohjelmoinnin opetuksessa tulee huomioida seuraavat asiat:

- tulee valita yksinkertainen ohjelmointikieli (esimerkiksi Python), jotta oppilaat voivat keskittyä syntaksiin
- tulee valita huolellisesti ratkaistaviksi useita sellaisia tehtäviä, jotka ovat ohjelmointikielestä riippumattomia ja kehittävät algoritmista ajattelua
- tulee opettaa soveltuvien ohjelmointikielten ja –ympäristöjen avulla.

Ohjelmoinnin, kuten matematiikankin, opetuksessa ja oppimisessa on merkityksellistä, missä järjestyksessä eri asiat opetetaan. Selby (2015) ehdottaa ohjelmoinnin opetuksen tehtävän seuraavassa järjestyksessä:

1. rakenteet, faktat, tyypit
2. miten erillinen rakenne toimii
3. ohjelmarakenteiden käyttö suunnittelun yhteydessä
4. erittely, abstraktio
5. ohjelmien luominen, algoritmien suunnittelu
6. testaus, arviointi

Robins, Rountree ja Rountree (2003) ovat luoneet “ohjelmoinnin viitekehykset” (kuva 14) kuvaamaan yhteenvetona ohjelmoinnissa liittyvien asioiden yhteyttä toisiinsa. Viitekehyksen toisella akselilla ovat henkilökohtaiset ominaisuudet, nimittäin tieto, strategiat ja mentaaliset mallit. Kehyksen toinen akseli sisältää ohjelmoinnin eri vaiheet, suunnittelun, tuottamisen ja arvioinnin.

	Tieto	**Strategiat**	**Mallit**
Suunnittelu	suunnittelumenetelmät algoritmien suunnittelu viralliset menetelmät	suunnittelu ongelmien ratkaisu algoritmien suunnittelu	ongelmaympäristö kuvitteellinen ratkaisu
Tuottaminen	ohjelmointikieli ohjelmointikirjastot ohjelmointiympäristö / työkalut	algoritmien implementointi koodaus tiedon hakeminen	haluttu ohjelmisto
Arviointi	virheiden etsintätyökalut ja menetelmät	testaus virheiden etsintä seuranta virheiden korjaaminen	todellinen ohjelmisto

Kuva 14. Ohjelmoinnin viitekehys (Robins, Routree, Routree, 2003). Viitekehys kuvaa ohjelmointiin liittyvien asioiden suhdetta toisiinsa. Kuvaa tulee lukea pääasiassa sarakkeittain: *Tieto* suunnittelumenetelmistä (tarvitaan ohjelman *suunnitteluun*), *tieto* ohjelmointikielestä (tarvitaan ohjelman *tuottamiseen*), *tieto* virheiden etsintätyökaluista (tarvitaan ohjelman *arviointiin*), jne.

6.3 Ajattelumallien opettaminen

Ajattelun taidot ja menetelmät on sisällytetty uuteen opetussuunnitelmaan ja ne mainitaan matematiikan opetuksen keskeisissä sisältöalueissa. Ajattelumalleista puhuttaessa voidaan erottaa algoritminen ajattelu (algorithmic thinking) ja laskennallinen ajattelu (computational thinking).

Futschek (2006) on määritellyt algoritmisen ajattelun eräänlaiseksi joukoksi taitoja, joita käytetään algoritmien luomiseen ja ymmärtämiseen. Tämä joukko sisältää kyvyt

- analysoida annettu ongelma
- määritellä ongelma tarkasti
- löytää riittävät perustoimenpiteet ongelman ratkaisemiseksi
- rakentaa toimiva algoritmi perustoimenpiteitä käyttämällä
- ajatella ongelman mahdollisia erikois- ja normaalitapauksia
- parantaa algoritmin tehokkuutta.

Guzdial (2011) on ehdottanut, että tulisi hyväksyä laskennallisen ajattelun (computational thinking) määrittelyn olevan erittäin laaja. Olisi tärkeämpää miettiä, miten laskennallista ajattelua tulisi opettaa ja miten havaitaan, onko sitä opittu sen sijaan, että mietittäisiin laskennallisen ajattelun määritelmää. Selby (2015) on käyttänyt määritelmää, jonka mukaan laskennallinen ajattelu sisältää erittelyn (decomposition), abstraktion (abstraction), algoritmin suunnittelun (algorithm design), yleistämisen (generalization) ja arvioinnin (evaluation). Erittely on ongelman jakamista pienempiin osiin, jotka ovat helpommin ratkaistavissa kuin iso kokonaisuus. Abstraktio on kyky nähdä, mitkä ongelman osat ovat tärkeitä ja mitkä voidaan jättää huomiotta (Wing, 2008). Yleistäminen on kyky kuvata ongelman ratkaisu yleisin termein, joita voidaan soveltaa myös muihin samankaltaisiin ongelmiin. Ohjelman suunnittelun osana tarvitsee tehdä valintoja ja arvioida esimerkiksi ajan, virran tai ohjelmamuistin kulutusta. Tällaisen arvioinnin, jota voidaan laajentaa käyttöliittymiin tai prosesseihin, nähdään olevan yksi osa laskennallista ajattelua. Sykora (2014) määrittelee puolestaan laskennallisen ajattelun ongelmanratkaisuprosessina, joka sisältää seuraavat vaiheet:

- ongelman määrittely niin, että sen voi ratkaista tietokonetta tai muita työkaluja käyttämättä
- tiedon looginen organisointi ja analysointi
- tiedon esittäminen abstraktioiden, kuten simuloinnin, avulla
- ratkaisun toteutus huomioiden tehokkain tapa käyttää resursseja
- ratkaisuprosessin yleistäminen koskemaan useampia ongelmia

Ohjelmoinnin osaamista pidetäänkin tärkeänä ongelmanratkaisukykyjen ja loogisen päättelyn kehittymiselle. Tämän takia ohjelmoinnin integroimista osaksi opetusta kaikilla oppitasoilla pidetään kannattavana (Kalelioglu ja Gülbahar, 2014). Puhuttaessa 2000-luvulla tarvittavasta osaamisesta, Ornelas Marques ja Marques (2012) korostavat kriittisen ajattelun ja ongelmanratkaisukyvyn tärkeyttä. Heidän kokemuksensa mukaan oppilaat kehittävät juuri näitä kykyjä opetellessaan ohjelmointia. Heti ohjelmointiongelman kohdatessaan oppilaat tuntevat tarvetta ratkaista ongelman päästäkseen projektissa eteenpäin.

Lister ym. (2009) korostavat, että ohjelmoinnin oppimisessa on tärkeää oppia seuraamaan/lukemaan koodia ja selittämään, miten koodi toimii. Nämä vaiheet tulisi oppimisen hierarkiassa edeltää koodin kirjoittamista. Lee ym. (2011) ovat kehittäneet käytä-muuta-luo (use-modify-create) -mallin erityisesti laskennallisen ajattelun opettamiseen. Sentance (2017) on hyödyntänyt sekä Lister ym. (2019) että Lee ym. (2011) tutkimuksia ja kehittänyt ohjelmoinnin opetusmallia edelleen muotoon ennusta-käytä-tutki(selitä)-muuta-tee/luo/suunnittele (PRIMM, predict-run-investigate(explain)-modify-make/create/design). Ajatuksena on käyttää tätä mallia tekemällä oppitunnit ja oppituntien järjestys seuraavien aktiviteettien mukaisesti (Sentance ym., 2019):

- Ennusta (predict) mitä koodi tulee tekemään
- Käytä (run) koodia ennusteen testaamiseksi
- Tutki (investigate) koodin rakenne
- Muuta (modify) koodia lisäämällä toiminnallisuutta
- Tee (make) uusi ohjelma käyttäen samaa koodirakennetta.

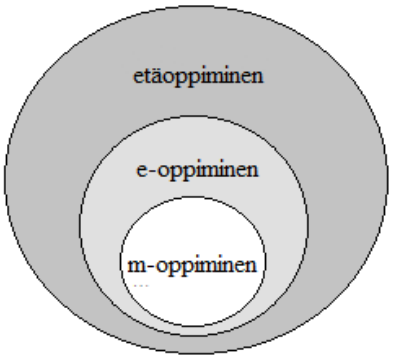
Tässä mallissa oppilaille annetaan ensiksi toimiva koodi ja heidän tulee ennustaa, miten se toimii (ennusta). Seuraavaksi ohjelmaa käytetään ja testataan, pitikö ennustus paikkansa (käytä). Tämän jälkeen ohjelmakoodin toiminta tutkitaan rivitasolla (tutki). Sitten ohjelmaa muutetaan tekemään jotain uutta (muuta). Viimeisessä vaiheessa suunnitellaan uuden ongelman ratkaisemiseksi uusi ohjelma, johon voidaan käyttää osia alkuperäisestä ohjelmasta.

7 Teknologian kehityksen vaikutus ohjelmoinnin oppimateriaaleihin

Perinteisessä luokkahuoneopetuksessa opettaja esittää oppimateriaalin oppilaille. Huolimatta opettajan ja oppilaan kohtaamisesta ja tällä tavalla annettavan palautteen selvistä hyödyistä on tällaisella opetuksella myös haittapuolia. Esimerkiksi mikäli oppilaalla ei ole mahdollista osallistua oppitunnille, saattaa hän menettää osan opetuksesta eikä saa kaikkea samaa opetusmateriaalia, joka käytiin tunneilla läpi. Tällaiset haitat ovat johtaneet siihen, että ollaan alettu etsimään uusia ja tehokkaampia opetusmenetelmiä. (Georgiev ym., 2004)

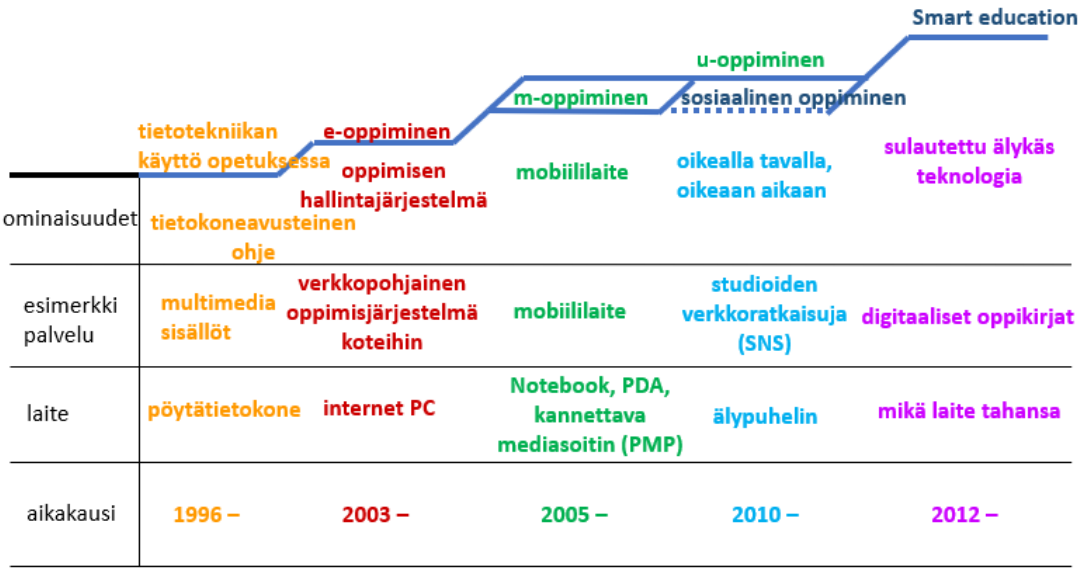
Vaikka yläkoulun ohjelmoinnin opetuksen on vaikea kuvitella tapahtuvan pääsääntöisesti etäopiskeluna, voi tällaisten opiskelumuotojen yhteydessä käytettävien teknologioiden kehityksellä olla suuriakin vaikutuksia ohjelmoinnin opetuksen oppimateriaalien kehittymiselle. Etä- ja pelioppimisessa käytettäviä teknologioita voidaan soveltaa hyvin myös pääosin luokkahuoneissa tapahtuvaan opetukseen. Tässä luvussa esitellään näiden teknologioiden kehittymistä ja esitellään miten tämä kehitys voi mahdollisesti vaikuttaa ohjelmoinnin opetuksen oppimateriaaleihin.

Teknologiaa voidaan hyödyntää opetuksessa sekä opiskelussa monilla eri tavoilla. Erityisesti informaatioteknologian, tietokoneiden, erilaisten päätelaitteiden ja tietoliikenteen tekninen kehitys sekä yleistynyt käyttö ja osaaminen ovat mahdollistaneet monia uusia teknologian käyttötapoja opetuksen yhteydessä. Näitä on sovellettu erityisesti etäopiskelussa. Tähän liittyviä yleisesti käytettyjä termejä ovat etäoppiminen (engl. d-learning), e-oppiminen (engl. e-learning) eli elektroninen oppiminen ja m-oppiminen (engl. m-learning) eli mobiilioppiminen. Kuvassa 15 on esitetty Georgiev ym. (2004) kuvaus näiden eri oppimistapojen suhteista toisiinsa. M-oppiminen on osa e-oppimista, joka on edelleen osa etäoppimista.



Kuva 15. M-oppimisen suhde e-oppimiseen ja etäoppimiseen (Georgiev ym., 2004).

Jang (2014) kuvaa m-oppimista seuraavana askeleena u-oppimisen (engl. u-learning, ubiquitous learning) eli ubiikin oppimisen. Kuvassa 16 hän on kuvannut eri oppimisvaiheiden käyttöönottoa etelä-korealaisessa opetuksessa. Viimeisimpänä vaiheena on ns. smart-oppiminen tarkoittaen itseohjautuvaa, motivoitunutta, muuntuvaa, resurssirikasta ja teknologiaan sulautettua (lyhenteenä englanninkielisistä sanoista self-directed, motivated, adapted, resource enriched, technology-embedded) oppimista.



Kuva 16. Oppimisen eri vaiheet etelä-korealaisessa opetuksessa. (Jang, 2014)

Teknologian käyttöä on hyödynnetty laajalti ohjelmoinnin opettamisessa ja tästä alueesta on myös tehty paljon tutkimuksia. Ohjelmointitaitojen oppiminen ei ole helppoa ja vaatii, että oppilas suorittaa ison määrän harjoitustehtäviä. Erityisesti vikojen etsintä ja korjaaminen voi olla vaikeaa, aikaa vievää ja altista inhimillisille virheille (Lam ym., 2008).

Rongas, Kaarna ja Kalviainen (2004) ovat jakaneet ohjelmoinnin opetuksessa käytettävät työkalut neljään kategoriaan: 1) integroitu kehitysrajapinta, 2) visualisointi, 3) virtuaalinen oppimisympäristö ja 4) järjestelmät harjoitusten palauttamiseen, hallintaan ja testaukseen. Verdú ym. (2011) mukaan on tärkeää, että ohjelmoinnin opiskelijoilla on käytössään vähintään virtuaalisen oppimisympäristön ja harjoitusten palauttamisjärjestelmän yhdistelmä. Virtuaalinen oppimisympäristö tarjoaa tuen ongelmien ratkaisuun ja itsenäiseen opiskeluun. Useat harjoitusten palautusjärjestelmät tukevat erilaisia harjoitustehtävä- ja arviointimuotoja, tarjoten myös mahdollisuuden automaattiseen tehtävien tarkistukseen ja palautteen antamiseen.

7.1 E-oppiminen

E-oppiminen ja elektroninen oppiminen ovat synonyymejä. Näillä termeillä tarkoitetaan mitä tahansa tietotekniikkaa hyväkseen käyttävää opiskelua (Meisalo, Sutinen, Tarhio, 2003). E-oppiminen voi sisältää itsenäisen, johdetun tai yhteisöllisen lähestymistavan oppimiseen (Blezu, Popa, 2008). Itsenäisessä e-oppimisessa jokainen oppilas suorittaa oppimiseen liittyvät aktiviteetit itsenäisesti omassa ympäristössään ja omalla aikataulullaan. Johdettu e-oppiminen on tarkoitettu suoritettavaksi vuorovaikutuksessa ohjaajien kanssa. Yhteisöllisessä e-oppimisessa oppilaat työskentelevät yhdessä verkkoympäristössä. Blezu ja Popa (2008) erottavat toisistaan myös synkronoidun ja epäsynkronoidun e-opetuksen. Synkronoidussa opetuksessa oppilaat etenevät samaan tahtiin, kun taas epäsynkronoidussa opetuksessa oppiminen voidaan suorittaa eri aikoihin. Johdettu ja yhteisöllinen e-oppiminen voivat olla synkronoitua tai epäsynkronoitua. Itsenäinen e-oppiminen on aina epäsynkronoitua. Markus (2008) mainitsee e-oppimisen trendeinä muutok-

sen perinteisestä opetuksesta joustavaan itseohjautuvaan opetukseen, muutoksen tuotesuuntautuneesta oppimisesta prosessisuuntautuneeseen oppimiseen sekä yhteisöllisen oppimisen.

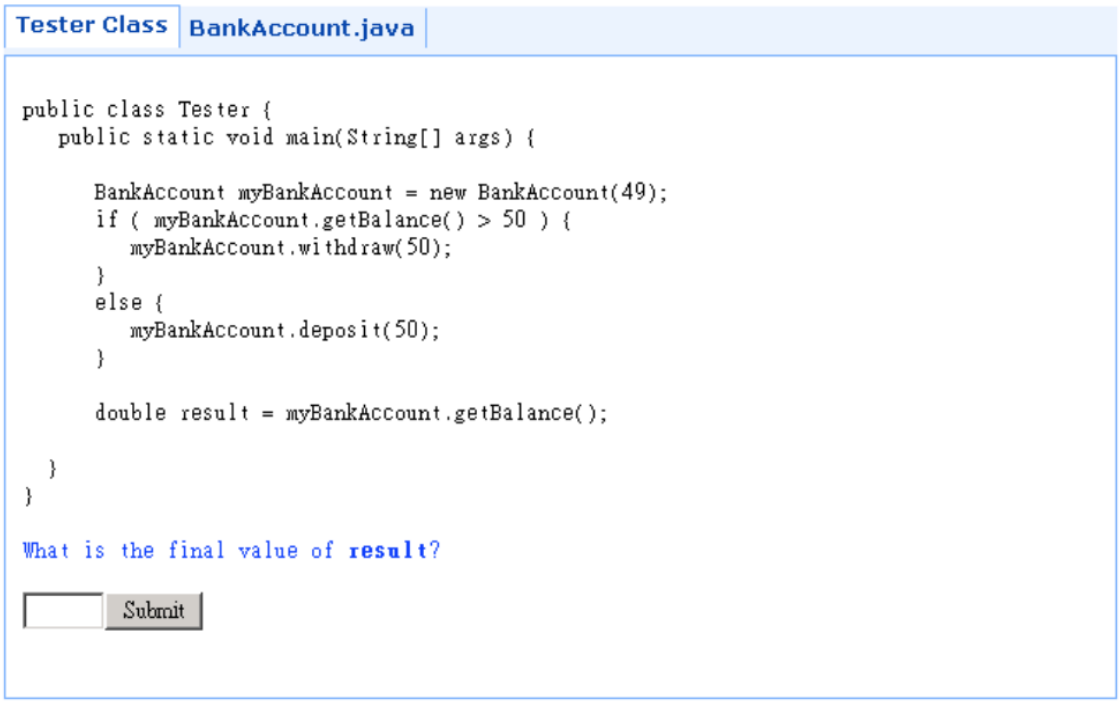
E-oppimisessa käytetään monia eri teknologioita, Blezu ja Popa (2008) listaavat esimerkkeinä seuraavat:
blogi, yhteisöllisen oppimisen ohjelmisto, tietokoneperusteinen arviointi, keskustelualusta, sähköposti, koulutuksen hallintajärjestelmä, koulutuksellinen animaatio, arvioinnin tukijärjestelmä, e-portfolio, peli, hypermedia, oppimisen hallintajärjestelmä, podcast, simulaatio, web-pohjainen opetusmateriaali, tekstikeskustelu, virtuaalinen luokkahuone, internet-sivu, web 2.0 -yhteisö ja wiki. Kaiken kaikkiaan voidaan todeta, että e-oppimisen oppimateriaaleissa voidaan hyödyntää todella monia erilaisia teknologioita.

Blezu ja Popa (2008) antavat kolme esimerkkiä e-oppimisen käytännöistä:

- web-pohjainen harjoittelu (web-based training), erityisesti yrityksissä käytössä oleva harjoittelumuoto, jossa netissä käytävä kurssi käydään ilman vuorovaikutusta (tai tukea) ammattiopettajien tai muiden työntekijöiden kanssa. Tämän tyyppisestä e-oppimismuodosta on kasvanut merkittävä toimiala.
- tuettu verkko-oppiminen (supported online learning), jossa suurin osa kurssin sisällöstä saatetaan toimittaa luentojen tai etäopetuksen kirjallisena materiaalina, mutta opetus luokitellaan e-oppimiseksi, koska yhteydenpito ohjaajaan, keskustelu muiden oppilaiden kanssa, tutkimusmateriaalin etsintä, yhteistyössä tehtävät harjoitukset, kurssin yleiskuvaukset ja tukimateriaali toimitetaan kaikki verkossa.
- vapaamuotoinen e-oppiminen (informal E-learning), johon luokitellaan työpaikoilla lisääntyvät teknologian käyttömahdollisuudet vapaamuotoisen oppimisen tukemiseksi edellä mainittujen “kurssipohjaisten” lähestymistapojen ulkopuolella.

Modernit e-oppimisympäristöt sisältävät interaktiivisuutta pelkkien staattisten sivujen sijaan. Interaktiivisten kysymysten tiedetään olevan osallistavia ja hyödyllisiä e-oppimisympäristössä. Itsearviointitilassa ne auttavat oppilasta ne puolestaan auttavat opettajia ohjaamaan oppilaita ja arvioimaan heidän edistymistään. (Hsiao ym., 2010)

Useat e-kurssit sisältävät suuren määrän erilaisia kysymyksiä, mutta niiden tarjonnasta ei saada täyttä hyötyä, jollei oppilaita ohjata valitsemaan juuri heidän osaamistasolleen sopivia kysymyksiä. Hsiao ym. (2010) ovatkin kehittäneet Java-ohjelmoinnin opetukseen QuizJET-ohjelmiston, joka sisältää suuren määrän interaktiivisia kysymyksiä sekä älykkyyttä ohjaamaan oppilaille juuri sopivan tasoisia kysymyksiä. Kuvassa 17 on näytetty esimerkki tämän ohjelmiston sisältämästä kysymyksestä, jolla testataan oppilaan kykyä ymmärtää Java-koodin toimintaa.



Kuva 17. Esimerkki QuizJET-ohjelmiston kysymyksestä (Hsiao ym., 2010)

7.2 M-oppiminen

Yksi muoto e-oppimisesta on m-oppiminen eli mobiilioppiminen, joka korostaa esimerkiksi matkapuhelimien ja tablettien sulautumista osaksi oppimisympäristöä (Meisalo, Sutinen, Tarhio, 2003). Georgiev, Georgieva ja Smrikarov (2004) määrittelevät m-oppimisen kykynä opiskella kaikkialla mihin aikaan tahansa ilman kiinteää fyysistä yhteyttä kaapeliverkkoon. Tämä voidaan tehdä käyttämällä kannettavia laitteita, kuten kämmentietokonetta (PDA), matkapuhelinta, kannettavaa tietokonetta tai tablet-tietokonetta.

Kukulska-Hulme ym. (2007) kirjoittaa tutkijoiden olevan yhtä mieltä siitä, että mobiilioppiminen on erilaista kuin muu teknologialla tuettu oppiminen. Siitä, mikä on merkittävin ero, on kuitenkin erilaisia näkemyksiä. Cortez ym. (2004) korostavat oppilaiden mahdollisuutta olla tehokkaammin yhteistyössä toistensa kanssa hyödyntämällä teknologian tuomaa mahdollisuutta luonnolliseen liikkuvuuteen. Vavoula ym. (2007) mainitsevat ettei mobiilioppimisessa ole enää kyse vain uuden teknologian käytöstä vaan uusien toimintojen tekemisestä niiden avulla. Opettaja voi suunnitella uudenlaisia oppikokemuksia mobiililaitteiden avulla rakentaen yhteyden teknologioiden, sisältöjen, kokemusten ja oppimisen välille. Yksi mobiilioppimisen tutkimusalue on se, että miten ajasta ja paikasta riippumatonta opiskelua voidaan tukea. Esimerkiksi Yau ja Joy (2006) ovat pyrkineet rakentamaan kontekstitietoisen oppijärjestelmän, jossa oppilaalle valitaan erilaisia tehtäviä riippuen muun muassa siitä, millaisessa paikassa (linja-auto, juna, kahvila, kirjasto) hän kulloinkin on.

Opettajille, joilla on vain vähän kokemusta m-oppimisesta, on helpointa käyttää olemassa olevia oppimateriaaleja ja toimittaa ne pienemmälle laitteelle sopiviksi. Monet oppilaat ovat tyytyväisiä, että materiaalit ovat saatavilla tällä tavalla ja käyttävät myös erilaisia kommunikointitapoja, jotka mobiililaitteet mahdollistavat. Tämä on kuitenkin vain yksi osa m-oppimista. M-oppimisen pitäisi tähdätä innovointiin ja löytämään, miten voidaan hyödyntää sitä, että käytettävissä on kannettavia laitteita, jotka tukevat havainnointia, kanssakäymistä, keskustelua ja pohdiskelua useissa eri tyyppisissä yhteyksissä. (Kukulska-Hulme ym., 2007)

7.3 U-oppiminen

U-oppiminen eli ubiikki oppiminen (engl. u-learning, ubiquitous learning) perustuu sulautettuun tietotekniikkaan (engl. ubiquitous computing). Sulautettu tietotekniikka on ympäristöönsä sulautuvaa, kaikkialla olevaa tietotekniikkaa, kuten sensoreita, RFID (radio frequency identification)-lappuja ja -kortteja, langattomia kommunikointilaitteita, matkapuhelimia ja puettavia tietokoneita. (Yahya, Ahmad & Jalil, 2010) Päälle puettavia tietokoneita ovat esimerkiksi kellot ja rannekkeet, joilla voidaan mitata mm. käveltyjen askelten määrää, pulssia tai unen laatua; silmälasit, joihin voidaan heijastaa erilaista tietoa tai kengät, jotka toimivat navigaattoreina led-valojen avulla (Richmond, 2013).

Hwang (2006) kuvaa u-oppimisympäristön sisältävän seuraavat ominaisuudet:

1. Ympäristö on kontekstitietoinen (context-aware) eli tietoa oppilaasta ja hänen ympäristöstään voidaan saada sensoreiden avulla.
2. Ympäristö antaa aktiivista, yksilöllistettyä tukea oikealla tavalla, oikeassa paikassa ja oikeaan aikaan perustuen henkilön ja ympäristön tilanteisiin sekä oppilaan profiiliin.
3. Ympäristö mahdollistaa jatkuvan opiskelun kaikkialla ja milloin tahansa eli opiskelu ei keskeydy oppilaan siirtyessä paikasta toiseen.
4. Ympäristö kykenee muokkaamaan oppiaineen sisältöä riippuen käytettävien mobiililaitteiden ominaisuuksista.

Taulukossa 2 on verrattu m- ja u-oppimisjärjestelmiä toisiinsa tuoden esille sen, miten ne eroavat toisistaan.

Taulukko 2. M- ja u-oppimisjärjestelmien erot (Hwang, 2006).

m-oppimisjärjestelmä	**u-oppimisjärjestelmä**
Järjestelmä ymmärtää oppilaan tilanteen tietokannan avulla.	Järjestelmä ymmärtää oppilaan tilannetta paitsi tietokannan kautta myös havaitsemalla oppilaan sijainnin sekä henkilökohtaiset ja ympäristön tilanteet todellisessa maailmassa.
Oppilaat käyttävät järjestelmää aktiivisesti langattomien verkkojen kautta.	Järjestelmä tarjoaa aktiivisesti yksilöllisiä palveluita oppilaille heidän kontekstinsa perusteella.
Oppimisportfolio tallentaa oppilaan verkkokäyttäytymisen.	Oppimisportfolioon kirjataan paitsi verkkokäyttäytyminen myös oppilaan reaalimaailman käyttäytyminen ja ympäristötiedot.
Järjestelmä tarjoaa tukea, joka perustuu vain oppilaan profiiliin ja tietokannan online-käyttäytymiseen.	Järjestelmä tarjoaa henkilökohtaista tukea oikealla tavalla, oikeaan aikaan perustuen oppilaan henkilökohtaisiin ja ympäristön tilanteisiin
Järjestelmä tarjoaan ‘milloin ja missä tahansa’ -opiskelun vain määrätyssä ympäristössä tietyille mobiililaitteille. Laitteiden vaihto tai siirtyminen keskeyttävät opiskelun.	Järjestelmä tarjoaa oppimisen missä ja milloin tahansa, vaikka oppilas liikkuu paikasta toiseen ja ympäristö (mukaan lukien laitteet ja verkot) muuttuu.
Oppilaan on asennettava ohjelmistot tietyille mobiililaitteille.	Järjestelmä mukauttaa kohteen sisältöä aktiivisesti vastaamaan erilaisten mobiililaitteiden toimintoja.

7.4 Pelioppiminen

Pelioppiminen (engl. game-based learning) tarkoittaa videopelien käyttöä opettamisen ja oppimisen tukemiseksi (Perrotta ym., 2013). Pelioppiminen on kehittynyt merkittävästi viimeisten vuosikymmenten aikana, sillä pelipohjaisissa oppimisympäristöissä on onnistuttu yhdistämään tehokkaita ongelmanratkaisukeinoja erittäin kiinnostaviin oppimiskokemuksiin (Lester ym., 2013). de Freitasin & Griffithsin (2008) mukaan pelitekniikat ja -käytännöt ovat tulossa yhä yleisemmiksi jokapäiväisessä sosiaalisessa kanssakäymisessä ja pelien kehitys onkin tänä päivänä edelläkävijä erilaisten medioiden yhdistämi-

sessä. Esimerkkinä tällaisesta teknologisesti edistyksellisestä koulutuspelistä he mainitsevat usean pelaajan verkossa pelattavat roolipeli, joissa pelaaminen yhdistetään simulaatioon, kuten America's Army-peli.

Digitaaliset pelit voivat olla oppimistyökaluja, motivaattoreita ja uteliaisuuden herättäjiä ja sitä kautta tehokkaita välineitä optimoimaan oppilaan oppimista päivittäisessä opetustyössä. Useissa tutkimuksissa onkin vahvistettu positiivinen yhteys oppimisen ja oppilaan sitoutumisen välillä digitaalisia pelejä pelatessa (Papadakis, 2008). Ken (2008) mukaan tutkijoiden yleisesti käyttämät perustelut pelien käyttämiselle ovat:

- pelit kannustavat oppilaita aktiiviseen osallistumiseen
- pelit kannustavat aktiiviseen oppimiseen
- pelit voivat olla tehokkaita työkaluja laajentamaan oppimisprosessia erityisesti monimutkaisissa ja epämääräisissä aiheissa
- pelit edistävät oppilaiden välistä yhteistyötä.

Gee (2003) antaa seuraavat esimerkit siitä, miten hyviä oppimisperiaatteita on sisällytetty hyviin peleihin:

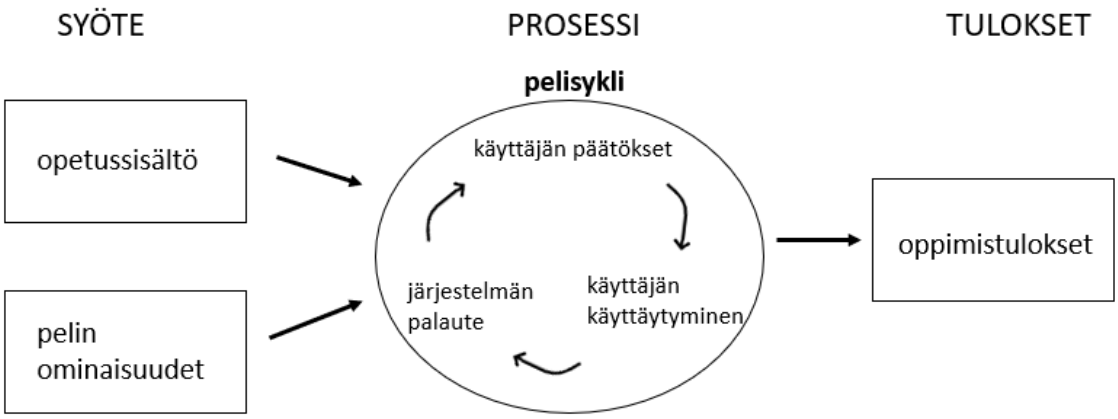
1. Hyvät pelit antavat tietoa pyydettäessä ("on demand") ja oikea-aikaisesti ("just in time") ja niin, että tieto ei ole irrallaan sen hetkisestä käytöstä tai pelaajan tarpeista tai tavoitteista. Esimerkkinä System Shock 2 -peli, jossa jaetaan koko pelin ajan sellaista tietoa, mikä yleensä löytyy ohjekirjoista.
2. Hyvät pelit toimivat pelaajan osaamisen rajalla, ollen haastavia, mutta tehtävissä olevia. Esimerkkinä Rise of Nations -peli, jossa pelaajat voivat säätää pelin osia ja peli tarjoaa tasotestejä varmistaakseen, että pelaajat löytävät omien taitojensa rajat.
3. Pelaajat voivat olla tuottajia eivätkä pelkästään kuluttajia. Esimerkkinä Rise of Nations ja Star Wars: Knights of the Old Republic -pelit, joissa pelaajat itse toimivat pelien rakentajina.
4. Hyvät pelit haastavat pelaajat pelin alussa ongelmilla, jotka ovat erityisesti suunniteltu antamaan pelaajille käsityksiä siitä mikä tulee toimimaan hyvin, kun he

kohtaavat vaikeampia ongelmia. Esimerkkinä Half-Life -peli toistaa samantyyppisiä ongelmia, kunnes pelaaja saavuttaa rutiinitason niiden ratkaisemiseksi. Tämän jälkeen pelaajaa haastetaan uudenlaisia ongelmien ratkaisemisella.

Kiili (2004) pitää oppimispeleissä tärkeänä, että peli antaa pelaajalle välitöntä palautetta, selvät tavoitteet ja haasteita, jotka sopivat hänen taitotasoonsa. Kim & Ko (2017) vertasivat tutkielmassaan erilaisia koodauksen opetuksen verkkoympäristöjä ja totesivat oppimispelien tarjoavan monia kontekstimuotoja, jotka saattavat auttaa opiskelijoita osallistumaan harjoitteluun aktiivisesti. He kehuivat myös pelien antamaa välitöntä ja henkilökohtaista palautetta ja suosittelivat esimerkiksi pelien Gidget, Lightbot ja Code Hunt käyttöä opetuksessa.

de Freitas ja Griffiths (2008) tuovat esille sen, että ohjaajille on erityisen haastavaa, koska on epäselvää, mitkä pedagogiset strategiat ovat tehokkaimpia ja kuinka vuorovaikutusta voidaan tukea pelejä käytettäessä. Tilannetta vaikeuttaa osaltaan se, että pelien yhdistyminen muihin teknologioihin on uutta ja luonteeltaan nopeasti muuttuvaa.

Garris ym. (2002) mukaan on olemassa oppimismalli, joka kuuluu suurimpaan osaan opetuspelejä. Tavoitteena on suunnitella opetusohjelma, joka sisältää tiettyjä pelien ominaisuuksia. Nämä ominaisuudet saavat aikaan syklin, joka sisältää käyttäjän päätökset, käyttäytymisen sekä järjestelmän antaman palautteen. Mikäli onnistutaan yhdistämään opetussisältö sopivien pelitoimintojen kanssa, tästä seuraa toistuva ja itsemotivoituva pelaaminen. Lopuksi tämä sitoutuminen johtaa oppimistavoitteiden saavuttamiseen. Tämä oppimismalli on esitetty kuvassa 18.



Kuva 18. Pelioppimisen oppimismalli (Garris ym., 2002)

Prensky (2005) nostaa esille kaksi syytä, miksi digitaalisia pelejä halutaan ja tulee käyttää opetuksessa:

1. Oppilaat ovat muuttuneet radikaalisti. Digitaalisen teknologian mukana kasvaneiden lasten älyllinen tyyli ja mielet eroavat heidän vanhempiensa vastaavista.
2. Nämä oppilaat pitää motivoida uusilla tavoilla.

Prensky (2005) näkikin digitaalisiin peleihin perustuvan oppimisen olevan vuonna 2005 vastaavassa tilanteessa kuin autoteollisuus oli vuonna 1890, lentoteollisuus 1910 ja tietokoneet 1950.

8 Tutkimustehtävä

Tutkimustehtävässä kartoitetaan yläkoulun matematiikan opettajien kokemuksia, suunnitelmia ja ideoita ohjelmoinnin opetuksesta osana matematiikan opetusta. Tarkoituksena oli sekä nykytilanteen kartoitus että opettajien kokemusten pohjalta kehittämisideoiden kerääminen.

Kysely lähetettiin 70 pääkaupunkiseudun yläkoulujen rehtoreille ja he välittivät kyselyn koulunsa matematiikan opettajille. Vastauksia saatiin 34 opettajalta, mitä voidaan pitää hyvänä määränä ottaen huomioon kyselyn ajankohta. Nimittäin kysely tehtiin sen lukuvuoden aikana, jolloin uusi opetussuunnitelma oli juuri otettu käyttöön ja monissa kouluissa käytettiin, esimerkiksi säästösyistä, vielä vanhan opetussuunnitelman mukaisia oppikirjoja.

8.1 Tutkimusmenetelmä

Tutkimusmenetelmäksi valitsin kyselytutkimuksen, joka toteutettiin sähköisenä kyselynä. Kysely toteutettiin Google Forms -alustaa käyttäen (www.google.com/forms/about/). Muita mahdollisia tutkimustapoja tällaisen tiedon keräämiseksi olisi esimerkiksi paperinen kyselytutkimus tai haastattelut joko kasvotusten tai puhelimitse. Olen käyttänyt samaa tutkimusmenetelmää ‘Opettaja työnsä tutkijana’ -kurssin harjoitustyössä (Haapakangas, 2019) ja seuraavaksi esitellyt perustelut tämän menetelmän käyttämisestä, eduista ja haitoista ovat samat kuin ko. työssä.

Evansin ja Mathurin (2005) tekemän tutkimuksen mukaan web-kyselyllä on selviä etuja muihin tutkimustapoihin verrattuna. Näitä etuja ovat muun muassa joustavuus, nopeus ja täsmällisyys, helppokäyttöisyys, tiedon syötön ja analysoinnin helppous, kysymysten monimuotoisuus, seurannan helppous ja tietämys vastaajista ja ei-vastanneista. Tekemäni tutkimus eroaa tuosta tutkimuksesta siten, että minulla oli pienempi ja selkeästi määritelty vastaajaryhmä (pääkaupunkiseudun yläkoulun matematiikan aineenopettajat), kun taas Evansin ja Mathurin tekemä tutkimus koski internetissä tehtävää kyselyä, jonka vastaajaryhmää ei oltu tarkoin määritelty.

Mahdollisina web-pohjaisen kyselytutkimuksen heikkouksina Evans ja Mathur (2005) mainitsevat muun muassa roskapostiksi tulkitsemisen, vastaajan web-taidot, epäselvän ohjeistuksen, yksityisyyden ja alhaisen vastausmäärän. Tässä tutkimuksessa näitä heikkouksia pyrittiin hallitsemaan seuraavilla tavoilla:

- roskapostiksi tulkitseminen: kyselyyn liittyvän sähköpostin otsikko ja viestin alku olivat selkeitä ja kuvaavia, jotta viestiä ei sekoitettaisi roskapostiin.
- vastaajan web-taidot: koska vastaajaryhmänä olivat opettajat, jotka käyttävät työssään tietotekniikkaa, ei web-taitojen puutteellisuus ollut relevantti ongelma tässä kyselyssä.
- epäselvä ohjeistus: sähköpostiin liitetty ohjeistus oli lyhyt ja selkä.
- yksityisyys: kyselyyn sai halutessaan vastata nimettömänä.
- alhainen vastausmäärä: niille, jotka eivät olleet vastanneet määräaikana tai olivat vastanneet nimettöminä, lähetettiin muistutussähköposti kyselystä.

Koska ohjelmoinnin opetus on määritelty opetussuunnitelmassa toteutettavaksi osana matematiikan opetusta, valittiin kyselyn kohderyhmäksi yläasteen matematiikan opettajat. Kysely lähetettiin pääkaupunkiseudun matematiikan opettajille sähköisesti kevään 2018 aikana. Osa kysymyksistä olivat monivalintakysymyksiä ja osa avoimia kysymyksiä.

8.2 Tutkimuskysymykset

Kysely sisälsi kaiken kaikkiaan 20 kysymystä, joista osa oli monivalintaisia, osa avoimia. Kysymykset oli jaoteltu seuraaviin aihealueisiin: taustatiedot, ohjelmoinnin opetus käytännössä, koulun teknologiaympäristö, oppimateriaalit ja kokemuksia ohjelmoinnin opettamisesta. Kysymykset on esitetty liitteessä yksi.

Pääpaino kysymyksissä oli oppimateriaaleissa, niiden vahvuuksissa, puutteissa ja parannusehdotuksissa.

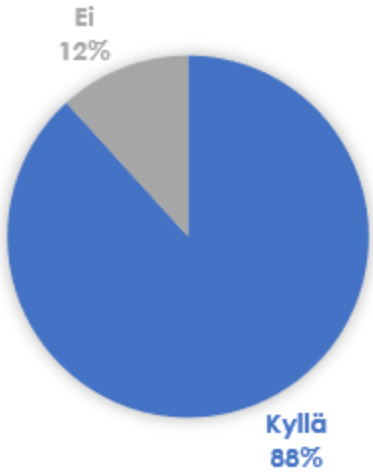
8.3 Tutkimustulokset

Seuraavissa kappaleissa esitellään opettajien vastauksia tutkimuskysymyksiin. Vastaukset ovat jaoteltu kyselyn mukaisiin ryhmiin. Joissakin tapauksissa annetut vastaukset sopivat paremmin toiseen kysymykseen kuin siihen, minkä vastauksina ne on annettu. Tällaisissa tapauksissa vastauksia on siirretty sen kysymyksen alle, mihin ne parhaiten sopivat. Osa vastauksista on jätetty pois, jos ne eivät tuo lisäarvoa jo näytetyille vastauksille tai eivät vastaa suoraan esitettyihin kysymyksiin.

8.3.1 Ohjelmoinnin opetus

Näissä kysymyksissä kartoitettiin, opettivatko kyselyyn vastanneet opettajat ohjelmointia kyseisenä lukuvuonna ja olivatko he ohjelmoineet aikaisemmin.

Kysymys 1: Opetatko ohjelmointia lukuvuonna 2017–2018?
Kysymykseen saatiin 34 vastausta, joista 30 vastasi ‘kyllä’, 4 ‘ei’. Vastaukset on esitetty kuvassa 19.



Kuva 19. Ohjelmoinnin opetus vastaajien keskuudessa.

Syiksi sille, ettei ohjelmointia opetettu vastattiin mm. seuraavaa:

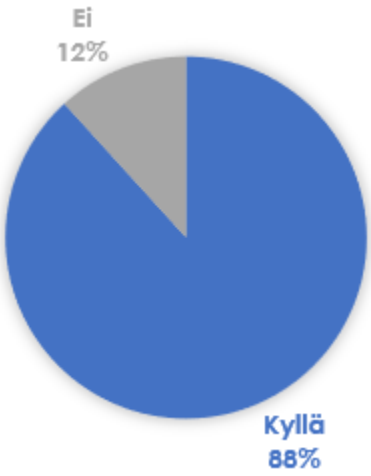
> *Ei ole tarjolla suoranaista ohjelmointikurssia. Toki ohjelmointia otan oppiaineiden ohessa, kuten fysiikassa.*

Uusi OPS voimassa vasta 7.lk. En opeta ko. luokka-astetta tänä lukuvuonna.

En ole saanut siihen koulutusta. Kirjasarjamme on vanha, jossa ei ole ohjelmointia.

Muuta opetettavaa on aivan tarpeeksi.

Seuraavaksi kysyttiin opettajien aikaisempaa kokemusta ohjelmoinnista. 30/34 vastaajaa oli ohjelmoinut aikaisemmin, vastaukset on esitetty kuvassa 20.

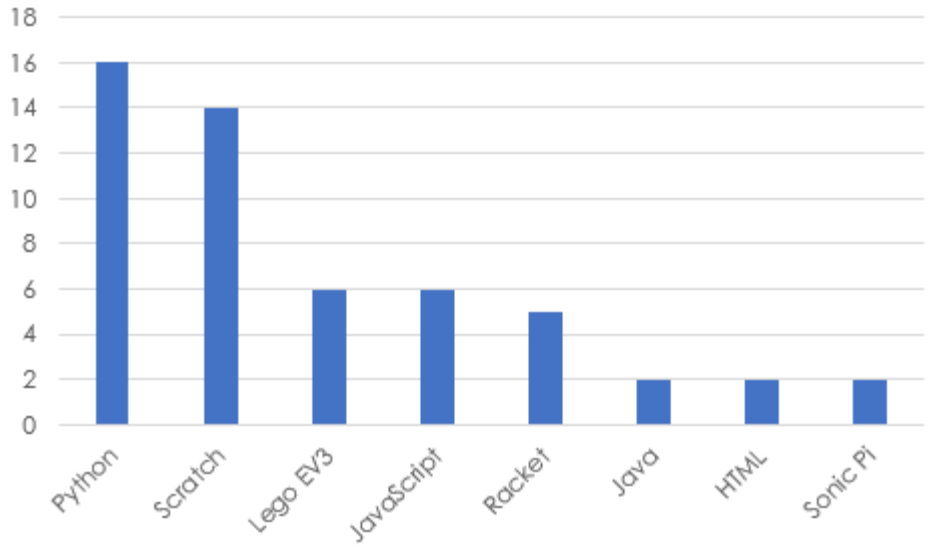


Kuva 20. Vastaajien aikaisempi kokemus ohjelmoinnista.

8.3.2 Ohjelmointikielet ja -ympäristöt

Näissä kysymyksissä kartoitettiin sitä, mitä ohjelmointikieliä ja millaisia ohjelmointiympäristöjä opettajat olivat suunnitelleet käyttävänsä ohjelmoinnin opetukseen.

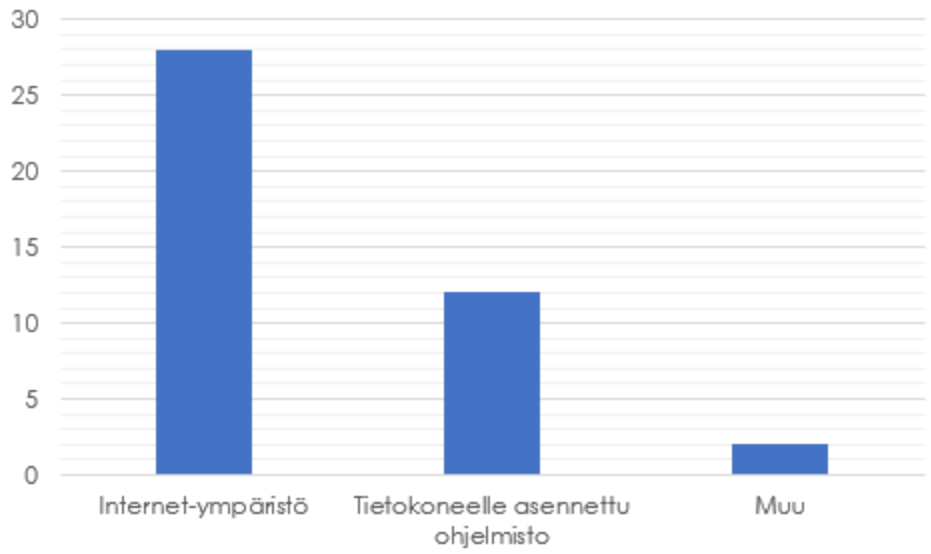
Kuvassa 21 on esitetty, mitä ohjelmointikieliä suunniteltiin käytettävän.



Kuva 21. Ohjelmoinnin opetuksessa käytettävät ohjelmointikielet

Kuten kuvasta 21 voi nähdä opetuksessa suunniteltiin käytettävän monia eri ohjelmointikieliä. Useat opettajat suunnittelivat käyttävänsä, tilanteen mukaan, erilaisia ohjelmointikieliä. Suosituin yksittäinen ohjelmointikieli oli Python, jota suunnitteli käyttävänsä noin kolmannes (14/34) vastaajista. Myös graafiset ohjelmointikielet Scratch ja Racket olivat suosittuja. Roboteista Lego Mindstorms oli suosituin, sitä suunnitteli käyttävänsä 6 eri vastaajaa.

Kuvassa 22 on esitetty, mitä ohjemointiympäristöjä opettajat aikoivat käyttää opetuksessa.



Kuva 22. Ohjelmoinnin opetuksessa käytettävät ohjelmointiympäristöt

Ylivoimaisesti suosituin ohjelmoinnin opetusympäristö oli kyselyn mukaan Internet-ympäristö, jota suunnitteli käyttävänsä 28 opettajaa, joista 15 ainoastaan Internet-ympäristöjä. Tietokoneelle asennettua ohjelmistoa suunnitteli käyttävänsä 12 eri opettajaa, joista 4 ainoastaan tietokoneelle asennettuja ohjelmistoja. Muita ohjelmistoympäristöjä suunnitteli käyttävänsä 2 opettajaa. Muina ohjelmistoympäristöinä mainittiin mm. Micro:bit ja Lego Mindstorms.

Seuraavaksi kysyttiin opettajien omia kokemuksia käyttämiensä ohjelmointikielten ja -ympäristöjen soveltuvuudesta opetukseen. Opettajilla oli saatujen vastausten mukaan kokemuksia erilaisten ohjelmointiympäristöjen käytöstä. Nämä kokemukset olivat pääosin myönteisiä ja monien eri ympäristöjen katsottiin soveltuvan hyvin ohjelmoinnin opetukseen. Kysymykseen vastattiin mm. seuraavaa:

Python-ohjelmoinnista sanottiin:

> *Sanoma Pron Python koodausympäristö. Ainakin alkupään tehtävät toimivat kohtuullisen hyvin.*
> *Pythonilla sekä arduinolla on tehty erinäisiä ohjelmia, joilla on kontrolloitu esim. ledin toimintaa.*

Scratchin soveltuvuudesta oltiin seuraavaa mieltä:

> *Erilaisia internet-ympäristöjä, kuten Scratch, Turtle Roy jne. Näin alkuvaiheessa on toiminut hyvin ja antanut oppilaille käsityksen loogisesta polkuajattelusta.*
>
> *Scratchilla on hyvä aloittaa ensikertalaisten kanssa, mistä edetään Micro:bit -piireihin. Micro:bit:ejä ohjelmoitaessa saa luontevan siirtymän blokeista kirjoitettuun tekstiin.*
>
> *Scratch toimii hyvin monen ikäisille (myös yläkoulu) sekä matikan opetuksessa (esim. koordinaatisto), että tietenkin tietotekniikan.*
>
> *Scratch on oppilaille helppo käyttää ja on ymmärrettävä.*

Lego Mindstormsien käytöstä annettiin seuraavia kommentteja:

> *Pääasiallisesti yläkoululaisten kanssa ohjelmoidaan Legon EV3 ympäristössä. Se on erinomainen koodausta aloitteleville.*
>
> *Lego mindstormsin ympäristö on hyvä.*
>
> *Legorobotteja matematiikan opetuksessa; ohjelmoinnillisen ajattelun opettelussa. On toiminut hyvin, oppilaat ovat olleet innostuneita ja päässeet robottien kanssa helposti liikkeelle.*

Muista ohjelmointikielistä ja -ympäristöitä sanottiin:

> *Valinnaisessa tietotekniikassa Internet-ympäristöjä (Hour of Code, Codeavergers jne) ja ohjelmointipelejä erityisesti niille, jotka ovat olleet kiinnostuneita.*

Olen käyttänyt opetuksessa Racketia ja sen käyttö vastasi tavoitteita. Saimme tehtyä sen mitä halusimme.

Dr. Racket PC:lle asennettuna tai Wescheme:llä toimii ok.

repl.it tai vastaava. Ollut käytössä valinnaiskursseilla. Toimii riittävän hyvin.

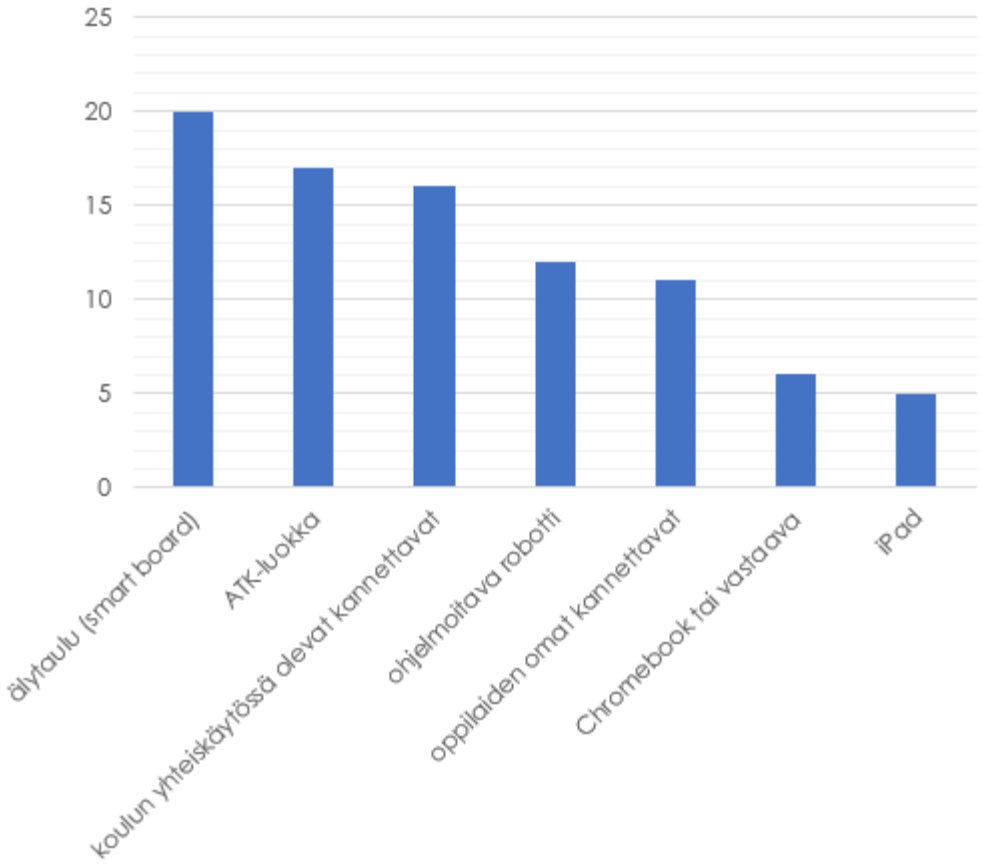
Kaikki eivät olleet tyytyväisiä, mutta tarkempaa tietoa ympäristöjen puutteista ei vastauksista saatu, vaan ne olivat enemmän korkealla tasolla, kuten

Mitään oikein sopivaa yläkouluun en ole vielä löytänyt.

8.3.2 Koulun teknologiaympäristö

Näissä kysymyksissä kartoitettiin sitä, millaiset tietotekniset laitteet opettajilla on käytettävissä ohjelmoinnin opetusta varten ja ovatko ne heidän mielestään riittävät.

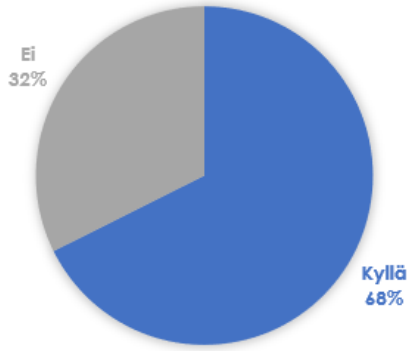
Kuvassa 23 on esitetty, mitä laitteita kouluilla on käytettävissä ohjelmoinnin opetukseen.



Kuva 23. Kouluilla käytössä olevat tietotekniikan laitteet.

Kuten kuvasta 23 voi nähdä kouluilla on käytössään erilaisia tietotekniikan laitteita. Suosituin yksittäinen tietotekniikan laite oli älytaulu, jonka 20 vastaajaa ilmoitti olevan koulullaan käytössä. Vastausten mukaan jonkinlaisia tietokoneita oli käytössä joka koulussa, eniten käytössä olivat koulun ATK-luokka ja koulun yhteiskäytössä olevat kannettavat. Noin 1/3 oli käytössä myös erilaiset tablet-laitteet, lähinnä Chromebook tai iPad.

Seuraavaksi kysyttiin, kokivatko opettajat koululla olevat laitteet ja tarvikkeet riittäviksi ohjelmoinnin opetukseen. Kuten kuvasta 24 näkee, noin 2/3 vastaajista kokivat koulunsa laitteet ja tarvikkeet riittäviksi ohjelmoinnin opetukseen.



Kuva 24. Koulun laitteiden/tarvikkeiden riittävyys ohjelmoinnin opetukseen

Kysymykseen mitä koulun laitteiden osalta voisi parantaa saatiin vastauksina monia parannusehdotuksia. Yleisiksi ongelmiksi koettiin laitteiden riittämätön määrä ja tekniset puutteet:

> *Periaatteessa on riittävät. Käyttöaste vain on niin suuri, ettei laitteita riitä aina, kun haluaisi.*
>
> *Aina voisi olla enemmän laitteita.*
>
> *PC-tä saisi olla enemmän ja helpommin saatavilla.*
>
> *Läppäreitä ei välttämättä aina ole kaikille saatavilla, jos useat ryhmät haluavat käyttää niitä yhtä aikaa.*
>
> *ATK-luokka on kovin varattu.*
>
> *Jokainen luokkatila tarvitsisi useita koneita ja sähköpistokkeita ja toimivan netin.*
>
> *ATK-luokan pöytäkoneet ovat todella hitaita.*

1 atk-luokka eli 15-20 konetta per luokka-aste eli 100 oppilasta, ON liian vähän.

Kannettavissa ei ole erillisiä hiiriä.

Tällä hetkellä pöytäkoneet ovat todella vanhoja ja eivät sen takia toimi.

Myös laitteiden käyttöönotto koettiin joissakin tapauksissa haasteelliseksi:

Tarvikkeet riittävät, mutta uusien ohjelmien käyttöönotto on hankalaa ja hidasta, jos ohjelmiston tarvitsee asentaa. Ainoastaan alueelliset ATK-tuet voivat asennuksia koneille tehdä.

Ohjelmien asentaminen. Lego-roboteilla työskentelyssä haasteena on, että kaikilla koneilla ei ole sitä ohjelmistoa ja pyynnöistä huolimatta kaupungin ATK-tukihenkilö ei ole saanut asennettua sitä kaikkiin, koska siinä on joku tekninen ongelma.

Vaikka näillä mennään, niin hyvä lausekieli olisi tarpeen. Kollega yritti asentaa pythonin eikä siitä oikein tullut mitään vaikka mikrotuki kyllä auttoi. Koko kaupungin tasolla pitäisi olla paketit javaan, pythoniin ja vaikka scalaan ja vielä tuki niiden toimintakuntoon saattamiseen. Nyt kun oppilaille tulee omat kannettavat, paketit pitäisi saada myös heidän koneilleen. Netissä toimivat ympäristöt olisivat parhaita, kuten scratch.

Meille on tulossa oppilaille Chromebookit tammikuussa. Niihin ei voi asentaa mitään.

Joissakin vastauksissa toivottiin saatavan ohjelmoinnin opetuksen avuksi uusia, erilaisia laitteita, kuten robotteja ja elektroniikka-alustoja:

MicroBit, Arduino, Legorobotit yms. olisivat hyvä konkreettinen lisä.

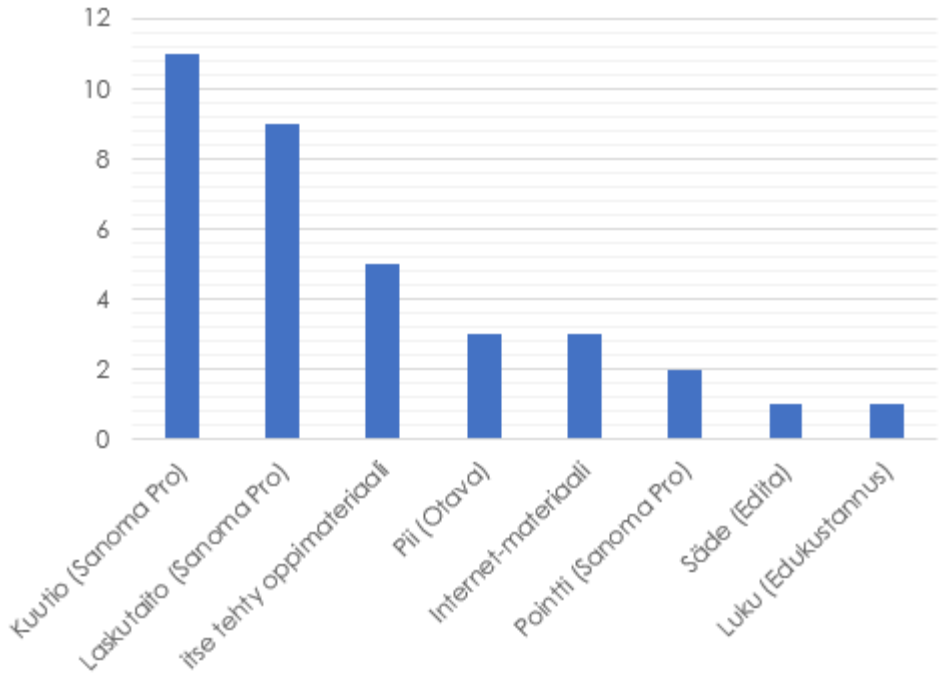
Aina laitteita voisi tietysti olla lisää. Alakoululaisille voisi olla esim. Microbit settejä.

Robotiikkaa olisi kiva saada myös yläluokille, nyt niitä on vaan alaluokilla.

Jos suuremmassa määrin kokeilee ohjelmointia, niin erilaisia hyviä sovelluksia olisi mukava saada, jossa perusohjelmointia voisi harjoitella.

8.3.3 Oppimateriaalit

Näissä kysymyksissä kartoitettiin sitä, millaisia oppimateriaaleja ohjelmoinnin opetuksessa käytetään osana matematiikan opetusta. 32/34 vastaajaa mainitsi käyttävänsä jonkun suomalaisen kustantamon matematiikan oppikirjaa. Kaksi vastaajaa mainitsi käyttävänsä itsetehtyjä materiaaleja, koska eivät olleet löytäneet markkinoilla olevista oppikirjoista itselleen sopivia materiaaleja. Kuvassa 25 on näytetty käytettyjen materiaalien jakauma.



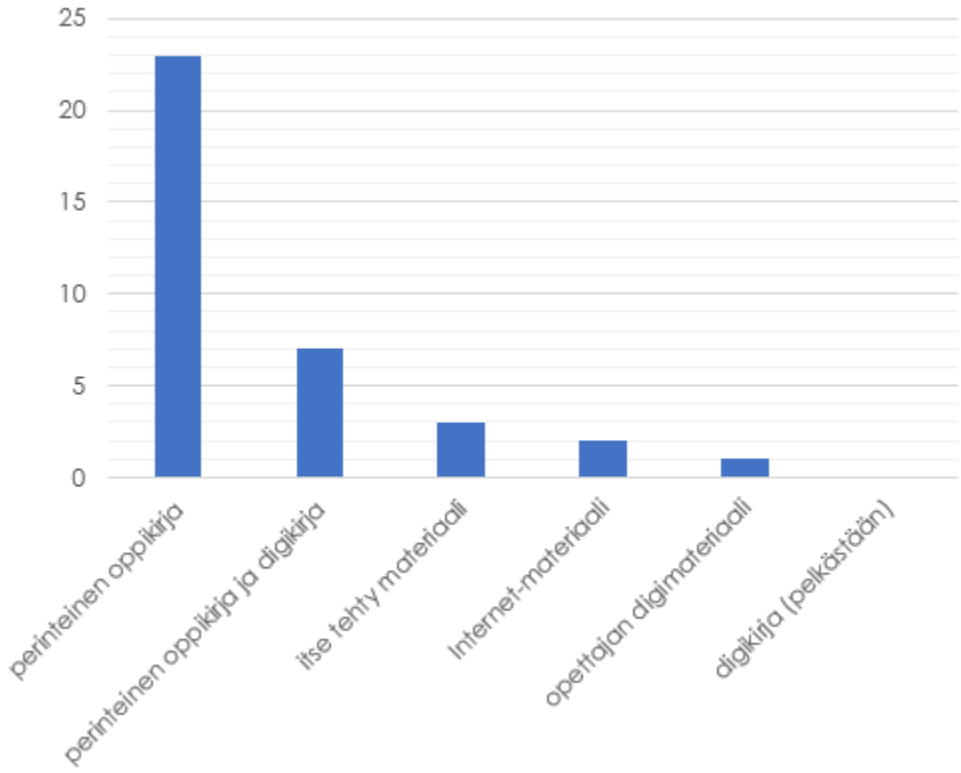
Kuva 25. Matematiikan opetuksessa käytettävät oppimateriaalit

Tavallisten oppikirjojen lisäksi vastaajat ilmoittivat käyttävänsä seuraavia materiaaleja:

> *Yleisesti verkon materiaaleja (Opetus.tv mm.), itse tehtyjä fyysisiä ja sähköisiä, pelejä, työpöytäohjelmia, ...*
>
> *Kahoot*
>
> *ViLLE sähköiset tehtävät, pulmapelit ja -kirjat*
>
> *Digitaalinen oppimispolku verkossa ja siihen omaa materiaalia*
>
> *Olen käyttänyt Peda-netistä löytyvää materiaalia opetuksessa.*

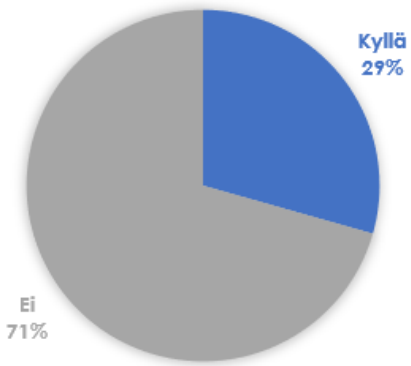
Seuraavaksi kysyttiin käyttävätkö opettajat opetuksessa perinteistä oppikirjaa, pelkkää digikirjaa vai jossain muussa muodossa olevaa oppimateriaalia. Kuten kuvasta 26 näkyy, noin 2/3 vastaajista käytti pelkkää perinteistä paperista oppikirjaa ja noin 1/4 perinteistä

oppikirjaa yhdessä digikirjan kanssa. Kukaan vastaajista ei ilmoittanut käyttävänsä pelkästään digikirjaa.



Kuva 26. Matematiikan opetuksessa käytettävät oppimateriaalityypit

Seuraavaksi kysyttiin, kokivatko opettajat ohjelmoinnin sisällytetyn hyvin käyttämäänsä oppimateriaaliin. Suurimman osan (24/30 vastaajaa) mielestä ohjelmointia ei oltu sisällytetty oppimateriaaliin tarpeeksi hyvin, kuten kuvasta 27 voi nähdä.



Kuva 27. Vastaajien kokemukset siitä, miten ohjelmointi on sisällytetty matematiikan oppimateriaaliin.

Opettajilta kysyttiin käyttävätkö he jotain muuta materiaalia ohjelmoinnin opetukseen matematiikan kirjan lisäksi. Monilla vastaajista oli kokemusta ohjelmoinnin opetuksesta ja he käyttivätkin monia erilaisia saatavilla olevia sekä itse valmistettuja materiaaleja.

Osa vastaajista kertoi tarkasti ohjelmoinnin opetuksessa käyttämästään materiaalista:

> *Innokas-verkoston materiaalit ovat olleet kovassa käytössä mm. "asimov" robottikirja. Lisäksi olen tuottanut itse materiaalia ohjelmoinnin tueksi.*
>
> *Tabletkoulun ohjelmointikirja yläkoulussa*
>
> *Koodiaapinen ja PedaNet Racket kielen materiaali*
>
> *MAOLin koulutuksessa läpi käytyjä materiaaleja.*
>
> *Koodausta kouluun -sivuston pohjalta tehtyä Maolin materiaalia*
>
> *code.org*
>
> *Linkki-keskuksen materiaalia*

Scratch ja sen ohjeita

Käytän vain tällä hetkellä Sanomapron Kuutio-sarjan sähköistä alustaa.

Legon tuottamia sisältöjä

Metropolian yläkoulun ohjelmointi -materiaalia liittyen Python-koulutukseen

Racket kurssin materiaaleja

Osa vastaajista kertoi käyttävänsä tarkemmin määrittelemättä netistä löytyvää ja/tai itse tekemäänsä materiaalia:

Olen laatinut geometriaan ohjelmointitehtävän itse MAOL:n ohjelmointikoulutuksessa jaettujen ideoiden pohjalta.

Olen opettanut ohjelmointia niin kauan, että lähinnä omaa päätäni käytän -- ja oppilaat tuovat aina jotain uutta ja ajankohtaista mukanaan.

Omaa kokemustani sekä käytyjä koulutuksia.

Omat materiaalit: fronteriin materiaalia, sekä kokeita että ihan teoriaa.

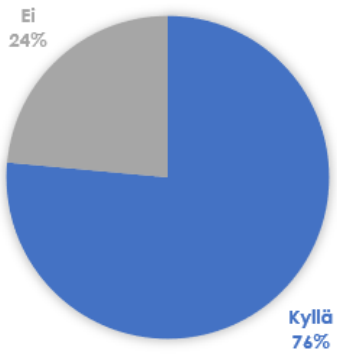
Omia ideoita ja joskus netistä löydettyjä materiaaleja.

Verkosta ja koulutuksesta saatuja materiaaleja

Netistä löydettyjä legorobottiohjelmoinnin oppaita. Internet-ympäristöjä käytän materiaalina opetuksessa.

Omat materiaalit, scratch ja micro:bit -materiaalit verkosta

Seuraavat kysymykset koskivat sitä, tukeeko heidän käyttämänsä ohjelmoinnin oppimateriaali opetussuunnitelmaan asetettuja tavoitteita ja miten kuntakohtaiseen opetussuunnitelmaan sisältyy ohjelmointi. Kuvassa 28 on esitetty opettajien kokemukset siitä, tukeeko ohjelmoinnin oppimateriaali opetussuunnitelmaan asetettujen tavoitteiden saavuttamista.



Kuva 28. Opettajien kokemukset siitä, tukeeko oppimateriaali OPS:n tavoitteiden saavuttamista.

Kuntakohtaiseen opetussuunnitelmaan ohjelmointi oli vastaajien mukaan sisällytetty seuraavilla tavoilla:

> *"Tavoitteet: ohjata oppilasta kehittämään algoritmista ajatteluaan sekä taitojaan soveltaa matematiikkaa ja ohjelmointia ongelmien ratkaisemiseen Sisältöalueet: Ohjelmoidaan ja samalla harjoitellaan hyviä ohjelmointikäytäntöjä. Sovelletaan itse tehtyjä tai valmiita tietokoneohjelmia osana matematiikan opiskelua."*
>
> *Sitä on itse asiassa hyvin vähän matematiikassa suhteessa siihen, minkä verran on muuta asiaa. Jos opettaja osaa tuoda esiin oppilaille, milloin*

itse asiassa käydään ohjelmointia (lausekkeen muodostamisen opettele-minen, yhtälönratkaisut, muuttuja- ja vakiokirjaimet), niin ohjelmointia on ikään kuin enemmän.

T20 ohjata oppilasta kehittämään algoritmista ajatteluaan sekä taitojaan soveltaa matematiikkaa ja ohjelmointia ongelmien ratkaisemiseen S1 Oh-jelmoidaan ja samalla harjoitellaan hyviä ohjelmointikäytäntöjä. Sovel-letaan itse tehtyjä tai valmiita tietokoneohjelmia osana matematiikan opiskelua.

Sisällytetty kyllä, mutta mitään ohjeistusta ei ole annettu tai materiaaleja toteutukseen. Lisäksi olisi ollut mukava päästä sellaiseen ideapäivään, jossa olisi saanut ideoita oppitunneille.

Olen ollut tekemässä kunnalle TVT opetussuunnitelmaa ja siinä on selkeä koodauspolku varhaiskasvatuksesta lukioon; eli on meidän kunnassa erinomaisessa kunnossa.

Kunnassa on annettu opettajille aika vapaat kädet. Racket-kielen koulu-tusta on järjestetty opettajille.

Epämääräisesti eli hyvin avoimeksi jää mitä tehdään, mutta ehkä hyväkin niin, jolloin voi itse päättää.

Valtakunnallisen OPS:n tavoitteet ja sisällöt vuosiluokkaistettu.

Matematiikan osana, myös meillä valinnaisessa at-oppiaineessa.

Lyhyen epämääräisesti kuten OPSissakin

Kuntakohtaisessa opsissa ohjelmointi on aikalailla samoin kuin valtakunnallisessa opsissa.

Kunnassamme on erikseen TVT-ops, jossa ohjelmoinnin tavoitteet näkyvät selkeästi.

On sisällytetty, mutta koulutusta ei ole riittävässä määrin järjestetty.

Seuraavaksi kysyttiin onko ohjelmoinnin oppimateriaaleissa puutteita ja mitä niissä voisi kehittää.

Osa vastaajista koki, että ohjelmoinnin oppimateriaaleissa ei ole puutteita:

Ohjelmointi on otettu huomioon uuden opsin mukaisissa kirjoissa, jokaisessa hieman eri tavoin. Ei näissä sinänsä puutteita ole, pintaraapaisuksihan ohjelmoinnin opettelu on tarkoitettukin. En osaa sanoa opettajien puolesta, jotka eivät ole ohjelmointiin ennen millään tavalla tutustuneet.

Ei ole vielä tullut vastaan. Tarkoituksenani on kuitenkin muovata materiaaleja oppisisältöihin sopivimmiksi.
Ainakin nyt alkuun materiaali on toiminut riittävän hyvin.

Ei ole puutteita
En osaa sanoa, käytän ensisijaisesti itse laatimiani materiaaleja ja täydennän muulla.

Joidenkin vastaajien mielestä puutteita oli käytettävyydessä:

Välillä epäloogisia etenemistapoja, vaikea navigoida eteenpäin.

Käytettävyyden toivon parantuvan.

Tehtävät eivät ole kovin konkreettisia ja niiden suoritusta on vaikea valvoa.

Automaattitarkistus yms. lisämaksullisia

Osa vastaajista kokivat oppikirjoissa olevat tehtävät puutteellisiksi tai irrallisiksi osiksi matematiikan opetukseen:

Oppikirjoissa olevat materiaalit ovat hyvin köykäisiä. Ei ole mielekästä tehdä vihkoon tehtävää, jossa tulkitaan jotain koodinpätkää. Sekä oppilaille että opettajalle voisi olla konkreettisempia oppimateriaaleja ohjelmoinnista.

Tehtävä eteviltä oppilailta loppuu äkkiä kesken.

Kunnon pitkäjänteisiä tehtäviä ohjeineen. Tehtävät voisivat olla itsestään eriyttäviä eli oppilas voisi oman tasonsa mukaan vaikeuttaa tehtävää.

Ohjelmointi on Kuutio-kirjassa yhtenä kappaleena keskellä kirjaa, ei sovi mielestäni tuohon kohtaan, koska tuntuu irralliselta.

En ole vielä nähnyt yhtään sellaista materiaalia, joka olisi luonteva osa esim. matematiikan tunnilla.

Käytettävässä kirjasarjassa sitä ei ole laisinkaan, joten materiaali on etsittävä muualta.

Oppikirjassa kyllä, mutta onneksi materiaalia löytyy netistä.

> *Ylipäätään laadukkaita oppimateriaaleja on vaikea löytää. Lisäksi ne vanhentuvat nopeasti.*

Seuraavaksi kysyttiin minkälaisia oppimateriaaleja ohjelmoinnin opetukseen opettajat toivovat. Tähän kysymykseen saatiin paljon vastauksia.

Moni vastaaja toivoi parempaa (digitaalista) nettimateriaalia. Tällaisen materiaalin katsottiin myös auttavan eriyttämisessä ja motivoimisessa:

> *Materiaaleja, joilla pääsee alkuun, vaikka oppilas ja opettaja eivät kumpikaan osaisi etukäteen MITÄÄN, mutta jotka tarjoavat silti vuosia koodanneille oppilaille riittävästi mielenkiintoista tehtävää. Eli SUURI eriyttämisen tarve.*

> *Selkeitä ohjeistukseltaan, itseohjaavaa, jolloin jokainen voi edetä tason mukaan. Eroja on valtavasti oppilaiden osaamisessa.*

> *Koodausympäristö, joka antaa selkeät ohjeet kunkin koodausvaiheen toteuttamiseen ja tarkastaa vastaukset kunnolla. Perusteluja ja keinoja, joilla motivoida oppilaita.*

> *Valmiita järkevästi mietittyjä hommia, joilla näkee erilaisia koodaustapoja. If-lauseille jne. jotain käteviä tapoja. Sopiva ohjelmoinnin harjoitteluympäristö esim. netissä, jossa on runsaasti tehtäviä eri vaikeusasteilla ja jossa selkeällä suomella käydään läpi perusteet ja mennään vähän syvemmälle. Ohjelmassa olisi hyvä olla jokin "palkitsemistasosysteemi", jolla voisi kilpailla toisten kanssa ja vaikka valtakunnallisesti "hall of fame" (koulun sisällä ja valtakunnallisesti).*

> *Valmiita tehtäväkokonaisuuksia, joiden teon voi aloittaa hyvin pienellä kynnyksellä ja jotka ovat itseohjaavia.*

Nettikurssi, joka vastaa yläkoululaisten taitotasoa. Sellaisia tehtäviä, joilla saa näkyvää aikaan, esim graafinen lopputulos tai ääni.

Toiminnallista nettimateriaalia. MOOC-tyylisiä omalla tahdilla eteneviä kursseja.

Eiköhän nämä ole ihan päteviä. Lukioissa matematiikan opiskelussa käytetään digitaalisia alustoja esim. moodle joten jokin tällainen voisi olla hyvä jo peruskoulussa tutustuttavaksi.

Monipuolisia tehtäviä, yksinkertaisista haastaviin

Meillä oli parikymmentä vuotta sitten käytössä ohjelmisto nimeltä Toolbook, joka oli tarkoitettu oppimateriaalin tuotantoon. Se oli hyvin samanlainen kuin Applen HyperCard. Siinä saattoi piirtää, tuoda kuvaa, tehdä painikkeita, ohjelmoida animaatiota, kyselyjä jne. Kaipaan sellaista kokonaisvaltaista työkalua, jossa oppilas voi itse päättää, käyttääkö ympäristöä graafiseen tarinankerrontaan vai kunnon ohjelmointiin.

Monissa vastauksissa esitettiin toivomuksia materiaalien sisällön suhteen sekä kommentoitiin myös sitä, miten ohjelmointia tulisi opettaa:

Oma kirja/materiaali ohjelmoinnin opetukseen. Ja toivoisin, että ohjelmointia opettaisi tietojenkäsittelyn opettaja!

Selkeän rungon siitä, mitä pitäisi opettaa, mistä lähteä liikkeelle, mitkä olisivat ne vaatimukset tai tavoitteet. Minimit saavutetaan ilman tietokonetta, jokaisella matikan kurssilla; maksimit liehuu jossain ohjelmointikielen opettelun oppimisessa. Ei hyvä. En pidä lainkaan siitä, että atk-opettajat on syrjäytetty ja matikkaan sisällytetty "koodaus".

Selkeitä tuntiohjeita, joiden avulla saisi pidettyä jäsennettyjä tunteja.

Muutaman tunnin kokonaisuuksia, kenties open ohjeita, kuinka sitä kannattaa opettaa.

Esim. Scratchista tms. voisi olla opettajille valmiita tuntisuunnitelmia.

Opetussuunnitelman tavoitteiden saavuttamisen tapa ja ohjelmointi opetuksen laajuus on vielä hieman epäselvää.

Jos nyt ensin joku kertoisi, että mitä sen ohjelmoinnin pitäisi olla yläkoulussa.

Oppimateriaaleja, jotka olisivat integroituneet jollain tasolla oppisisältöihin. Ts. Oppisisälllöt ovat uudessa opsissa melko laajoja. Jos ohjelmointi saataisiin osaksi uuden oppimista, päästäisiin tavoitteisiin molemmilla saroilla.

Oppimateriaaleissa voisi olla suoraan ohjelmointitehtäviä ja vinkkejä ohjelmoinnin opetukseen. Myös esim. opetusvideoita voisi olla suoraan tarjolla sekä oppilaille että opettajalle.

Helppokäyttöisiä, perusteet yleispätevästi opettavia

Perusasioista lähtevää, luokkatasot huomioon ottavaa suoraa ohjeistusta. Oppikirjojen ulkopuoleista, jonka tekijät uskaltavat kohdentaa peruskouluun ja tehdä kokonaisuuksia vuosiluokittain.

Yleistä oikeaan ajatteluun ohjaavaa, joka ei liity mihinkään tiettyyn kieleen.

Selkeästi lisää sellaista eri vuosiluokille suoraa suunnattua materiaalia, joka ei vaatisi myöskään opettajalta tuntikausien ennakko-opiskelua.

Riittävän käytännönläheisiä, helposti ymmärrettäviä ja hyvin motivoituja. Jos tehtävän voi suorittaa helpommin ilman ohjelmointia, tehtävä on huono.

8.3.4 Kokemuksia ohjelmoinnin opetuksesta

Seuraavaksi kysyttiin opettajien kokemuksia ohjelmoinnin opettamisesta ja millaisia haasteita he olivat kohdanneet.

Ajanpuute todettiin haasteeksi usean vastaajan mukaan:

Tässähän tuota on tutkailtu ja maisteltu, että millä tavalla opetusta lähtee sitten toteuttamaan. Ehkä ajallisesti on ollut haasteita, sillä joutuu itse selvittämään paljon enemmän, siirtelemään laitteistoja sekä sumplimaan millaisen ajan ja mistä välistä käyttää ohjelmointiin.

Aikaa ei ole tarpeeksi kaikkien oppilaiden neuvomiseen oppitunnin aikana, koska ovat tarvinneet paljon apua, vaikka opettajia olisi ollut kaksi paikalla. Aikaa ei ole ollut myöskään tarpeeksi monen oppitunnin käyttämiseen ohjelmoinnin opetukseen.

Siihen ei voi käyttää kovin paljoa aikaa, koska muut matematiikan opiskelun tavoitteet ovat tärkeämpiä.

Lisäksi haasteena on aika; olisi mahtavaa tehdä laajempia kokonaisuuksia, mutta ikävä kyllä matematiikan sisällöt vs. aika yläkoulussa on hyvin haasteellinen.

Ohjelmoinnin opetuksessa pitää tehdä jonkin verran valmistelua ja työtä, että pystyy lähtemään liikkeelle. Koska ohjelmointia ei sidottu mihinkään tiettyyn asiayhteyteen, niin voi olla hankalaa löytää sille sopivaa aikaa (vaikka näin ei tietysti pitäisi olla).

Oppilaat ovat kovin eritasoisia ja omat resurssit eivät riitä kaikkien neuvomiseen tarvittavissa määrin. Ohjelmoinnin opetukseen kunnolla menee paljon oppitunteja.

Oppikirjasarjojen materiaali on ihan riittämätöntä. Lisäksi kunnollisten koodaustuntien valmistelu vaatii runsaasti opettajan kouluttautumista omalla ajalla, jotta asiasta itsekin jotain ymmärtää ja osaa oppilaita neuvoa.

Osa vastaajista oli kohdannut teknisiä ongelmia ohjelmoinnin opetuksessa:

kaupunki ei toimita/anna lupaa käyttää tarvittavia materiaaleja

Tekniset ongelmat (koneet eivät toimi)

Koneet eivät toimi tai ole saatavilla silloin kuin pitäisi.

Ohjelmointiympäristö tökki aluksi, eli tunnusten kanssa oli ongelmia.

Laitteisto-ongelmia sekä hyvin kirjava osaamisen taso alussa

Osa vastaajista näki ohjelmoinnin oppimisen oppilaille erittäin vaikeana. Tähän kerrottiin erilaisia syitä, kuten resurssien puute tai englanninkielisen materiaalin ymmärtämisen vaikeus:

Osa ei vaan pääse kärryille. Oppilaat toivovat perinteisiä matematiikan-tehtäviä

Oppilaista vaikeaa

Kaikki oppilaat eivät osaa seurata ohjeita, koska ei viitsitä lukea. On helpompaa sanoa "mäentajuumitään". Kun kymmenen oppilasta ympärillä sanoo "mäentajuu", ja jokaista pitäisi auttaa kädestä pitäen, alkaa openkin motivaatio rakoilla. Kollegat ovat tahoillaan niin kiireisiä, että emme ehdi suunnitella opetusta yhdessä. Käytännössä yksi saa keksiä pyörän, ja muut kopioivat sen. Yhdessä työskentely olisi mielekkäämpää, mutta aikaa yhteissuunnitteluun ei ole.

Oppilaita ei kiinnosta ja he käyttävät tietokoneita vääriin tarkoituksiin. He eivät osaa/jaksa lukea ohjeita ja turhautuvat, kun eivät välittömästi tiedä miten edetä. 22 oppilaan ryhmän kanssa myös jokaisen oppilaan luokse ehtiminen on täysin mahdotonta, kun asia on uusi ja kaikki tarvitsisivat apua.

osalle oppilaista alku on kankeaa, kun ohjelmointi on niin erilaista kuin mihin on aiemmin totuttu

Ohjelmointi ei kiinnosta kaikkia, joten täytyy osata motivoida oppilaita tekemään sitä.

Luultavasti riittävä apu osoittautuu ongelmaksi. Ts. 18 oppilasta ja yksi opettaja.

Yksi opettaja, oppilailla paljon avun tarvetta.

Yläkoululaisen matematiikan taidot ovat vielä melko vähäiset. Ei meinaa löytyä sopivan tasoisia tehtäviä, että jokainen saisi ne tehtyä. Scratch on vähän lapsellinen. Oma aika ei meinaa riittää eri ympäristöjen tutustumiseen.

Nettiympäristöjen kanssa tällä hetkellä ongelma, ettei oikein oppilaiden englannin kielen taito riitä seuraamaan tehtävien antoa ja sitä miten toteutetaan.

Osa oppilaista valittaa ettei ymmärrä englanninkielistä materiaalia, jos käyttää tällä hetkellä netissä olevia pohjia. Joillekin ekoja kertoja koodatessa tuottaa vaikeuksia, ettei ihan jokainen rivi välttämättä tuota mitään näkyvää. Kuten aiemmin sanottu, niin suht vähällä kokonaisuudessaan ollut koodauksen opetus. Matematiikassa vähällä ja valinnaisessa tietotekniikassa enemmän.

Hyvien suomenkielisten nettipohjaisten ohjelmointiympäristöjen puute.

Opetussuunnitelman tapa sisällyttää ohjelmointi koettiin muutaman vastaajan mielestä haasteelliseksi:

Uusissa uuden OPS:n oppikirjoissa on ohjelmointi otettu huomioon. Koulumme käyttää kuitenkin vanhoja materiaaleja, koska kirjoja on paljon, eikä _matematiikka_ ole muuttunut. Opettaja ei opeta vain sitä mikä on oppikirjassa, vaan toteuttaa opetussuunnitelmaa.

OPS-teksti on liian väljä.

Kenelläkään ei tunnu olevan visiota siitä mitä ohjelmoinnilla tulisi saavuttaa peruskoulussa.

Mielestäni ohjelmoinnin opettaminen ei ole tarpeellista.

Mielestäni ohjelmointia ja yleensäkin atk:ta pitäisi opettaa erillisenä oppiaineena. Jos tuodaan jokin näin iso asia OPSiin, siihen pitää antaa myös aikaa!

Osa vastaajista oli kriittisiä oman osaamisensa suhteen:

Opettajan osaaminen tai paremminkin osaamattomuus

Oma ainehallinta ei riitä.

Oma osaaminen on heikkoa

oma ohjelmointikokemus on todella vähäistä ja 1990-luvulta, eikä ole muiden töiden vuoksi aikaa opetella ohjelmointia kunnolla.

Koulutuksen puutteen sekä ajanpuutteen asiaan perehtymisen ja suunnittelun kannalta.

Osa vastaajista ei ollut kohdannut haasteita opetuksessa. He kertoivat hyväksi toteamiaan käytänteitä ja materiaaleja:

Esim Turun yliopiston Ville -materiaali on toimiva

Oppilailla toimii mielestäni se, että perusasioihin löytyy joku ohje, vaikka malliohjelma, ja sitten he alkavat soveltaa sitä omiin ideoihinsa.

Scratchilla kun aloitin niin yllättävän mukavasti lähti, joten haasteita on ollut vähemmän kuin odotin.

9 Luotettavuus

Sarajärven & Tuomen (2018, 163-164) mukaan laadullisen tutkimuksen luotettavuuden arvioinnista ei ole yksiselitteisiä ohjeita, mutta tutkimuksen luotettavuutta voidaan arvioida seuraavien näkökulmien kautta: tutkimuksen kohde ja tarkoitus, oma sitoutumiseni tutkijana, aineiston keruu, tutkimuksen tiedonantajat, tutkija-tiedonantaja-suhde, tutkimuksen kesto, aineiston analyysi ja tutkimuksen raportointi. Tässä luvussa arvioidaan tekemäni tutkimuksen luotettavuutta näiden alueiden kautta.

Tutkimuksen kohde ja tarkoitus. Tekemässäni tutkimuksessa tutkittiin yläkoulun opettajien kokemuksia ja näkemyksiä ohjelmoinnin opetuksesta osana matematiikan opetusta. Tutkimuksessa painotettiin oppimateriaalien käyttöä ja etsittiin ideoita ja keinoja niiden parantamiseksi.

Oma sitoutumiseni tutkijana tässä tutkimuksessa. Katson oman sitoutumiseni lisäävän tämän tutkimuksen luotettavuutta. Minulla ei ollut juurikaan ennakko-odotuksia tulosten suhteen ja uskon pystyneeni tarkastelemaan tuloksia objektiivisesti.

Aineiston keruu. Tutkimus tehtiin kevään 2018 aikana. Ohjelmointi oli tullut opetussuunnitelmaan vasta vuoden 2016 syksyllä, joten vastaajilla ei ollut vielä pitkäaikaista kokemusta ohjelmoinnin opettamisesta osana matematiikan opetusta. Joissakin kouluissa oli vielä vanhan opetussuunnitelman mukaiset kirjat käytössä. Tutkimustulokset olisivat saattaneet olla joiltakin osin toisenlaisia, mikäli tutkimus olisi tehty myöhempänä ajankohtana.

Tutkimuksen tiedonantajat. Tutkimukseen valitsin tietyiltä alueilta yläkoulut, joiden rehtoreihin otin yhteyttä. He välittivät kyselyn koulunsa matematiikan opettajille. Vastauksia saatiin kaiken kaikkiaan 34 opettajalta. Kyselyyn vastanneista opettajista 88% opetti ohjelmointia kyseisenä lukuvuonna. Tuloksia analysoidessa huomasin, että ohjelmoinnin opetukseen liittyvät käytännöt poikkeavat paljon koulusta ja kunnasta riippuen. Mikäli tutkimukseen osallistujat olisivat joltain muualta alueelta Suomessa, poikkeaisivat tutkimustulokset todennäköisesti joiltakin osin nyt saaduista.

Tutkija-tiedonantaja-suhde. Tutkimukseen vastanneet opettajat kirjoittivat vastauksensa (halutessaan anonyymeinä) tehtyyn kyselylomakkeeseen. Annettuja vastauksia ei ole päivitetty jälkeenpäin.

Tutkimuksen kesto. Alkuperäinen vastausaika kysymyksille oli kolme viikkoa. Tätä jatkettiin useamman vastauksen saamiseksi lähettämällä muistutusviesti niille kouluille, joista ei tiedetty onko kyselyyn vastattu. Näin ollen tutkimuksen aineiston keruun kokonaisajaksi tuli viisi viikkoa.

Aineiston analyysi. Tehty kysely sisälsi sekä monivalinta- että avoimia kysymyksiä. Monivalintakysymysten vastauksille ei suoritettu erillistä analyysiä, vaan tulokset on esitetty sellaisinaan. Avoimilla kysymyksillä kerättiin tietoa opettajien kokemuksista, havaituista ongelmista sekä parannusehdotuksista. Vastauksia analysoidessa havaitsin, että monet mainituista ongelmista ja haasteista ovat koulu- tai kuntakohtaisia. Myös ohjelmoinnin opetuksen käytännöt poikkesivat hyvin paljon toisistaan koulusta ja opettajasta riippuen.

Tutkimuksen raportointi. Tutkimuksen luotettavuutta on pyritty lisäämään raportoinnin avoimuudella. Tutkimuksessa käytetyt kysymykset ovat esitetty tutkielman liitteessä siinä muodossa kuin ne kysyttiin. Vastaukset avoimiin kysymyksiin esitetty lähes alkuperäisessä muodossaan, ainoastaan muutamia kirjoitusvirheitä on korjattu sekä vastaajan anonymiteetti on pyritty suojaamaan poistamalla joitain anonymiteetin vaarantavia osia vastauksista. Muutama päällekkäinen ja kysyttyyn aiheeseen liittymätön vastaus on jätetty esittämättä.

Yhteenvetona pitäisin tehtyä tutkimusta luotettavana. Vastaajien taustat poikkesivat toisistaan, minkä takia vastauksiin saatiin monia erilaisia mielipiteitä ja näkemyksiä. Saatuja vastauksia voidaan pitää hyvinkin kattavina. On kuitenkin korostettava, että tulokset ovat sidottuja tiettyyn ajankohtaan. Vastaavan tutkimuksen tekeminen myöhemmin antaisi mitä todennäköisemmin joiltakin kohdin erilaisia tuloksia. Tämä pätee myös tutki-

muksen alueellisuuteen. Eri puolella Suomea ja eri kouluissa tai kunnissa on ohjelmoinnin opetuksen käytännöt poikkeavat toisistaan ja myös saadut tulokset osittain riippuvia siitä, missä kouluissa tutkimus tehdään.

10 Pohdintaa

Kysely tehtiin ajankohtana, jolloin uusi opetussuunnitelma oli juuri otettu käyttöön 7.luokalle. Joissakin kouluissa oli vielä käytössä vanhan opetussuunnitelman matematiikan oppikirjat. Tältä osin kaikilla vastaajilla ei ollut vielä kokemusta siitä, miten ohjelmoinnin opetus on sisällytetty uuden opetussuunnitelman mukaisiin oppikirjoihin. Toisaalta opetussuunnitelma koettiin monissa vastauksissa epämääräiseksi ja liian väljäksi eikä sen koettu sisältävän visiota siitä, mitä ohjelmoinnin opetuksella itse asiassa pitäisi saavuttaa peruskoulussa. Yhdessä vastauksessa kyseenalaistettiin koko ohjelmoinnin opetuksen tarve yläkoulussa. Opetussuunnitelman väljyydellä on hyvät ja huonot puolensa. Se mahdollistaa tilanteesta riippuen hyvinkin erilaisten opetustapojen valinnan, mutta toisaalta lisää myös eriarvoisuutta eri koulujen välillä, kun opetussuunnitelman ohjeita tulkitaan eri tavalla. Itse näkisin, että opetussuunnitelma voisi olla yksityiskohtaisempi ohjelmoinnin opetuksen tavoitteiden määrittelyn osalta.

Ohjelmointikielistä ja -ympäristöistä koskevasta kysymyksestä selvisi, että kunnasta ja koulusta riippuen ohjelmoinnin opetus on hyvin erilaista. Joissakin kouluissa käytettiin tekstipohjaisia ohjelmointikieliä, lähinnä Pythonia, kun taas osassa kouluista oli käytössä visuaaliset ohjelmointikielet, lähinnä Sctrach. Ohjelmoinnin opetusta on tehty ja tehdään myös valinnaisissa kursseissa, jolloin sitä ei ole liitetty matematiikan opetukseen. Näillä kursseilla oli myös hyvin erilaisia välineitä käytössä eri kouluissa. Yleisimmin käytössä olivat Lego-robotit, mutta myös elektroniikka-alustoja käytettiin useassa koulussa. Yhteenvetona voidaan todeta, että käytetyt ohjelmointikielet ja -ympäristöt poikkeavat hyvinkin paljon toisistaan koulusta riippuen ja oppilaat voivat olla hyvinkin eriarvoisessa asemassa ohjelmoinnin opetuksen suhteen riippuen siitä, missä kunnassa tai koulussa he opiskelevat. Käytetyistä ohjelmointiympäristöistä Internet-ympäristö oli suosituin. Internet-ympäristöjä on useita erilaisia eikä tässä tutkimuksessa ole selvitetty tarkemmin, mitkä ympäristöt olivat suosituimpia. Internet-ympäristöjen suosioon vaikutti todennäköisimmin käytettävissä olevat päätelaitteet ja se, että ei tarvinnut erikseen asentaa ohjelmistoja. Monissa kouluissa oli tablet- tai notebook-laitteita sekä joissakin tapauksissa tietokoneiden mainittiin olevan vanhentuneita.

Uusi opetussuunnitelma määrittelee ohjelmoinnin alkeet jo osaksi alakoulun opetusta ja alakoulun jälkeen tavoitteena on “oppilas osaa ohjelmoida toimivan ohjelman graafisessa ohjelmointiympäristössä” (Opetushallitus, 2014). Näin ollen oppilaat tulevat jatkossa tutustumaan ohjelmoinnin alkeisiin, lähinnä visuaalisia ohjelmointikieliä käyttäen, jo alakoulussa. Tästä johtuen mielestäni olisi luonnollista ja paremmin opetustavoitteita (erityisesti algoritmisen ajattelun tavoitteita) tukevaa, että yläkouluissa matematiikan opetuksen ohessa tapahtuva ohjelmoinnin opetus tehtäisiin tekstipohjaisia ohjelmointikieliä käyttäen. Luonnollisin valinta täksi kieleksi olisi mielestäni Python, joka on jo tällä hetkellä yleisimmin yläkoulujen ohjelmoinnin alkeisopetuksessa käytetty kieli Suomessa ja useissa muissa maissa. Myös ohjelmointikieliä vertailevat tutkimukset puoltavat Pythonin sopivuutta ensimmäiseksi opetettavaksi tekstipohjaiseksi ohjelmointikieleksi (Mannila & de Raadt, 2006). Opettajien vastauksissa mielipiteet graafisten ohjelmointikielten käytöstä yläkouluissa ohjelmoinnin opetukseen jakautuivat selvästi. Osa piti graafisia ohjelmointikieliä, lähinnä Scratchia ja Racketia, hyvin yläkouluun soveltuvina, kun taas osa vastaajista piti tekstipohjaisten kielten käyttöä parempana vaihtoehtona.

Valinnaisissa ohjelmointikursseissa näen positiivisena erilaisten ohjelmointiympäristöjen käytön, koska tällainen saattaa motivoida paremmin yläkoulujen oppilaita ohjelmoinnin ja laskennallisen ajattelun oppimiseen. Robottien ja elektroniikka-alustojen rinnalle nostaisin älypuhelinten ohjelmoinnin – se saattaisi motivoida oppilaita jopa paremmin, koska älypuhelimet ovat osa heidän jokapäiväistä elämäänsä (Wagner ym., 2013). Sopivan oppimateriaalin löytäminen niin robottien kuin elektroniikka-alustojen ohjelmointiin vaatii tänä päivänä paljon aktiivisuutta ja oma-aloitteisuutta opettajilta. Verkossa on paljon erilaisia materiaaleja,mutta nämä ovat lähinnä erilaisia harjoitustehtäviä, joita voidaan oppilailla suorittaa. Nämä materiaalit ovat usemmiten jonkun opettajan oman aktiivisuuden tuloksena valmistettuja tai osana yliopistojen tutkimusta. Parhaiten suomenkielistä materiaalia on saatavissa Lego-robottien ohjelmointiin ja jonkin verran elektroniikka-alustojen (lähinnä micro:bitin) käyttöön. Älypuhelinten ohjelmointiin on valitettavan vähän suomenkielistä opetusmateriaalia ja opettajat ovatkin enemmän riippuvaisia englanninkielisistä materiaaleista.

Koulujen teknologiaympäristöt vaihtelevat luonnollisesti koulukohtaisesti. Kysymykseen teknologiaympäristön riittävyydestä ohjelmoinnin opetukseen jopa 32 % vastasi ympäristöjen olevan riittämättömiä. Tätä pidän huolestuttavan korkeana lukuna. Tietokonelaitteiden määrää ei pidetty monissa tapauksissa riittävänä ja jotkut mainitsivat laitteden olevan vanhoja ja tästä syysttä epäkäytännöllisiä. Onnistunut ohjelmoinnin opetus vaatii sitä tukevan riittävän hyvän laitteiston ja ennen kaikkea riittävän määrän laitteita niin, että jokaisella oppilaalla on ohjelmointiharjoituksia tehtäessä käytössään oma tietokone. Tekstipohjaisten ohjelmointikielten ohjelmointi on kätevintä tietokoneelta, esimerkiksi tablet-laitteita käytettäessä koodin kirjoittaminen on turhan hankalaa. Muita ohjelmoinnin harjoitteluun liittyviä tehtäviä kuin koodausta voidaan tehdä myös tablet-, notebook- tai jopa älypuhelin laitteita käyttämällä. Kussakin koulussa saatavilla oleva teknologiaympäristö vaikuttaa siihen, millaisia ohjelmointiympäristöjä ohjelmoinnin opetukseen kannattaa valita. Mikäli oppilailla on käytettävissään tarpeeksi tehokkaita pöytätietokoneita, voidaan valita ohjelmointiympäristö, joka ladataan tietokoneelle, kuten NetBeans. Mikäli käytettävissä on lähinnä notebook- tai tablet-laitteita, on parempi valita joko verkossa (Internet) tai koulun pilviympäristössä toimiva ohjelmointiympäristö kääntäjineen. Tällaista ympäristöä voidaan käyttää verkkoyhteyden kautta, pienenä hankaluutena on mahdollinen erillisen näppäimistön puuttuminen laitteesta, jolloin varsinkin kirjoittamista vaativien tehtävien suorittaminen on hieman hitaampia kuin näppäimistöä käytettäessä.

Matematiikan opetuksen oppimateriaalina käytettiin valtaosin suomalaisten kustantamoiden oppikirjoja. Kaikissa näitä kirjoja käyttävissä kouluissa oli käytössä perinteiset paperiset kirjat, joissakin näiden lisäksi digitaaliset, mutta missään koulussa ei käytetty pelkkiä digitaalisia oppikirjoja. Jopa 71 % vastaajista koki, että ohjelmoinnin opetus ei ollut sisällytetty hyvin käytettäviin matematiikan oppikirjoihin. Niinpä ohjelmoinnin opetukseen opettajat käyttivät oppikirjojen lisäksi paljon itse valmistamaansa tai verkosta hakemaansa materiaalia. Tämä tuli ilmi lähes kaikissa vastauksissa. Yksikään opettaja ei maininnut käyttävänsä ohjelmoinnin opetukseen ainoastaan matematiikan oppikirjoja. Suosituinta verkkomateriaalia olivat Innokas-verkoston, peda.netin ja code.org sivustojen materiaalit. Sekä opettajien vastaukset että tekemäni kirjallisuustutkimus toivat esille

tiettyjä asioita, joita tämän päivän oppimateriaaleista puuttuu. Ohjelmoinnin harjoitustehtävien ei tulisi olla pelkkiä ohjelmointitehtäviä vaan, ennen kaikkea harjoitusta laskennallisen ajattelun omaksumiseen. Lisäksi tehtävien tulisi sisältää myös koodin lukemista ja selittämistä sekä valmiiden koodipätkien muokkaamista. Joidenkin kustantajien valitsema ratkaisu tehdä erillinen oppikirja ohjelmoinnin opetukseen on mielestäni positiivinen, koska ohjelmointi on aihealueena niin laaja. Opettajan avuksi olisi myös automaattinen vastausten palautus- ja käsittely-ympäristö. Tällaisen ympäristön tulisi olla monipuolinen ja sisältää niin oppilaiden edistymisen seurantaa kuin vastausten monipuolista tarkastamista ja palautteen antamista. Joidenkin osan vaatimuksista täyttävinä ympäristöinä (kuten code.org ja codecademy) puutteena nähtiin se, että ohjelmoinnin oppimateriaalit eivät olleet kunnolla integroituneet oppisisältöihin vaan olivat erillisinä tehtävinä. Oppimateriaalin tulisi ehdottomasti sisältää hyvät ja yksityiskohtaiset opettajan ohjeet, sillä esimerkiksi selkeiden tuntiohjeiden puute koettiin ongelmana monissa vastauksissa.

Monien opettajien vastauksissa korostettiin hyvän nettipohjaisen ohjelmointiympäristön puutetta. Lisäksi toivottiin nettipohjaisia kursseja, jotka vastaisivat yläkoululaisten taitotasoa. Kurssien tulisi olla itseohjautuvia ja muuntuvia oppilaiden eritasoisen osaamisen suhteen. Myös mahdollinen pelioppimisen käyttö nähtiin mahdollisuutena, niin opettajien vastauksissa kuin kirjallisuustutkimuksessa (Lester ym.,2013, de Freitas & Griffiths, 2008).

Monet opettajat nostivat vastauksissaan esille aikaongelman. Uudessa opetussuunnitelmassa ohjelmointi lisättiin osaksi matematiikan opetusta, mutta sisällöstä ei poistettu mitään eikä opetukseen käytettävää tuntimäärää lisätty. Näin ollen on koettu ongelmalliseksi löytää tarvittava aika ohjelmoinnin opetusta varten.

Tämän tutkimuksen tuloksia voi hyödyntää ennen kaikkea suunniteltaessa oppimateriaalia ohjelmoinnin opetukseen yläkouluihin. Yhteenvetona ohjelmoinnin oppimateriaalin kehittämiselle osana matematiikan opetusta listaisin seuraavat asiat:

1. Käytössä tekstipohjainen ohjelmointikieli, mieluimmin Python.

2. Ohjelmointiympäristö, joka sisältää tuen tehtävien automaattiselle palautukselle ja tarkistamiselle.
3. Harjoitustehtäviä, jotka sisältävät muutakin kuin koodausta, esimerkiksi koodin lukemista, korjaamista, selittämistä ja parantamista.
4. Opetettavaan aiheeseen integroituja harjoitustehtäviä.
5. Eriyttämisen mahdollistavia harjoitustehtäviä.
6. Opettajan materiaalia, joka sisältää tunneilla läpikäytävän aineiston sekä tuntisuunnitelmat.
7. Mahdollisesti erillinen ohjelmoinnin oppimateriaali (oppikirja).

Viimeisessä kohdassa mainitun erillisen ohjelmoinnin oppimateriaalin tulisi linkittää ohjelmoinnin opetus matematiikan oppisisältöihin. Tämän lisäksi materiaali voisi sisältää valinnaisia ohjelmoinnin opetukseen liittyviä kurssikokonaisuuksia robotiikan, elektroniikka-alustojen tai älypuhelimien ohjelmoinnin alueilta.

Tämän tutkimuksen jatkoksi voisi tutkia erilaisten oppimateriaalien toteutusta käytännössä tutkimalla niiden vaikutusta ohjelmoinnin oppimistasoon sekä keräämällä palautetta opettajilta ja oppilailta erilaisten materiaalien käyttökokemuksista. Tällaisia tutkimuksia voitaisiin tehdä esimerkiksi seuraavien materiaalien käytöstä:

- code.org
- Codecademy

Tietyiltä alueilta voitaisiin kehittää osana tutkimusta oppimateriaalia ja sen käyttöä. Tällaisia oppimateriaalialueita voisivat olla esimerkiksi

- älypuhelinten ohjelmointi
- elektroniikka-alustat
- robotiikka
- peliohjelmointi

Tässä tutkimuksessa ei tutkittu tarkemmin millaisia erilaisia Internet-ohjelmointiympäristöjä käytettiin. Näitä olisi mahdollisuus tutkia ja vertailla tarkemmin.

Tätä tutkimusta vastaavan tutkimuksen tekeminen muutaman vuoden kuluttua, kun opettajilla on enemmän kokemusta uuden opetussuunnitelman käytöstä, voisi tuoda mukanaan mielenkiintoisia tuloksia. Myös vastaavan tutkimuksen laajentaminen koko Suomea koskevaksi voisi tuoda uusia, erilaisia näkökulmia asian käsittelyyn. Yläkoulun ohjelmoinnin opetuksen oppimateriaalien vertaaminen muissa maissa käytettäviin materiaaleihin olisi myös mielenkiintoinen tutkimuskohde.

Lähteet

Abelson, H. 2009. App Inventor for Android. Google Research Blog, July 31, 2009.

Bau D., Gray J., Kelleher C., Sheldon J. & Turbak F. 2017. Learnable programming: blocks and beyond. Communications of the ACM 60, 6: 72–80

Ben-Ari, M. 1998.Constructivism in computer science education. In Proceedings of the 29th SIGCSE Technical Symposium on Computer Science Education, pages 257–261. ACM Press.

Benitti F.B.V. 2012. Exploring the educational potential of robotics in schools: a systematic review, Computers and Education, 58(3), 2012, 978-988.

Blezu C., Popa E. 2008. E-Learning and its Prospects in Education. 12th WSEAS International Conference on Computers, Greece.

Code.org, www.code.org/international/about. Katsottu: 12.1.2020.

Cortez, C., Nussbaum, M., Santelices, R., Rodriguez, P., Zurita, G., Correa, C., & Cautivo, R. 2004. Teaching science with mobile computer supported collaborative learning. 2nd IEEE International Workshop on Wireless and Mobile Technologies in Education.

Crawford, R. 1999. Teaching and learning IT in secondary schools: towards a new pedagogy? Education and Information Technologies, Volume 4, 49– 63.

Dillenbourg, P. 1999. What do you mean by collaborative learning?. P. Dillenbourg. Collaborativelearning: Cognitive and Computational Approaches., Oxford: Elsevier, pp.1-19, 1999.

Edita, www.editapublishing.fi/oppimateriaalit/perusopetus/kirjasarjat/sade. Katsottu: 10.1.2020.

Edukustannus, www.edukustannus.fi/ylakoulu/luku/. Katsottu: 10.1.2020.

e-Oppi, www.e-oppi.fi/. Katsottu: 18.2.2020.

ePressi, www.epressi.com/tiedotteet/kustannustoiminta/edukustannus-rynnistaa-oppi-kirjamarkkinoille.html. Katsottu: 18.2.2020

European Commission 2014. Vassiliou urges Education Ministers to help kids crack the code. Press Release 29 July 2014.

Felleisen M., Findler R.B., Flatt M., Krishnamurthi S., Barzilay E., McCarthy J. & Tobin-Hochstadt S. 2015. The racket manifesto. In 1st Summit on Advances in Programming Languages (SNAPL 2015). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

de Freitas, S. and Griffiths, M.D. 2008. The convergence of gaming practices with other media forms: what potential for learning? Learning, Media and Technology, volume 33.

Friesen, N. 2001. What are Educational Objects? Interactive Learning Environments 2001, Vol. 9.

Futschek, G. 2006. Algorithmic thinking: the key for understanding computer science. Mittermeir, R.T. (Ed.), ISSEP 2006, LNCS, 4226, 159–168.

García-Peñalvo, F. J., Hughes, J., Rees, A., Jormanainen, I., Toivonen, T., Reimann, D., Tuul, M., & Virnes, M. 2016. Evaluation of existing resources (study/analysis). Belgium: TACCLE3 Consortium.

Gee, J.P. 2003. What Video Games Have to Teach Us About Learning and Literacy. New York: Palgrave/Macmillan.

Georgiev T., Georgieva E. & Smrikarov A. 2004. M-Learning – a New Stage of E-Learning. International Conference on Computer Systems and Technologies - CompSysTech’2004, Rousse.

van Gorp, M., Grissom, S. 2001. An Empirical Evaluation of Using Constructive Classroom Activities to Teach Introductory Programming. Computer Science Education, 2001, Vol. 11, No. 3, pp. 247-260.

Guzdial, M. 2011. A definition of computational thinking from Jeannette Wing. Computing Education Blog.

Haapakangas, E. 2019. Ohjelmointi matematiikassa. Opetuksessa käytettävät opetustavat. Opettaja työnsä tutkijana -tutkielma, Helsingin yliopisto.

Hadjerrouit, S. 2008, Towards a Blended Learning Model for Teaching and Learning Computer Programming: A Case Study. Informatics in Education, 2008, Vol. 7, No. 2.

Heinonen, J.-P. 2005. Opetussuunnitelmat vai oppimateriaalit – peruskoulun opettajien käsityksiä opetussuunnitelmien ja oppimateriaalien merkityksestä opetuksessa. Helsingin yliopisto. Tutkimuksia 257.

Hsiao, I.H., Sosnovsky, S. and Brusilovsky, P. 2010. Guiding students to the right questions: adaptive navigation support in an E-Learning system for Java programming. Journal of Computer Assisted Learning, 26(4), pp.270-283.

Hsu, Y. C., Rice, K., & Dawley, L. 2012. Empowering educators with Google's Android App Inventor: An online workshop in mobile app design. British Journal of Educational Technology, 43, E1-E5.

Hwang, G. J. 2006. Criteria and Strategies of Ubiquitous Learning. Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (pp.72-77), Los Alamitos: IEEE Computer Society.

IEEE 1484.12.1-2002 2002. Draft Standard for Learning Object Metadata.

Isotalus, P., Jussila, J., Matikainen, J. 2018. Twitter viestintänä. Ilmiöt ja verkostot. Vastapaino, Tampere.

Jamali, H., Nicholas, D., Rowlands, I. 2009. Scholarly e-books: the views of 16,000 academics. New Information Perspectives, 61(1), 33–47.

Jang, S. 2014. Study on service models of digital textbooks in cloud computing environment for SMART education. International Journal of u-and e-Service, Science and Technology, 7(1), 73-82.

Johnson, J. 2003. Children, robotics and education. In Proceedings of 7th international symposium on artificial life and robotics (Vol. 7, pp. 16–21), Oita, Japan.

Kaila, E., Kaarto, H., Laine, H. 2019. Ohjelmoinnin peruskurssi yläkouluun. Oppimisanalytiikan keskus

Kalelioglu, F., Gülbahar, Y. 2014. The Effects of Teaching Programming via Scratch on Problem Solving Skills: A Discussion from Learners' Perspective, Informatics in Education Vol. 13, 2014, 33-50.

Kang, J. H., Everhart, N. 2014. School librarians and the mandated implementation of digital textbooks in Florida and South Korea: Exploring school context. School Libraries Worldwide Volume 20, Number 2, July 2014

Kankaanranta, M. 2015. Digitaaliset oppimateriaalit – suuntana oppimisen adaptiivisuus ja vuorovaikutteisuus. Digitaalinen oppimateriaali koulun arjessa, Jyväskylän yliopisto.

Ke, F. 2008. A case study of computer gaming for math: engaged learning from gameplay? Computers and Education, Vol. 51, No. 4, pp.1609–1620.

Kiili, K. 2004. Digital game-based learning: Towards an experiential gaming model. Internet and Higher Education.

Kilpailu- ja kuluttajavirasto, 2016. Päätös KKV/453/14.00.10/2016.

Kim, J. H-Y. & Jung, H-Y. 2010. South Korean Digital Textbook Project. Computers in the Schools.

Kim, A. S., & Ko, A. J. 2017. A pedagogical analysis of online coding tutorials. Proceedings of SIGCSE, 321-326.

Kukulska-Hulme, A., Traxler, J. & Pettit, J. 2007. Designed and user-generated activity in the mobile age. Journal of Learning Design Vol. 2 No.1.

Laakso, M., Kaila, E. & Rajala, T. 2018. ViLLE – collaborative education tool: Designing and utilizing an exercise-based learning environment. Education and Information Technologies, 23(4), pp. 1655-1676.

Lam, M.S., Chan, E.Y., Lee, V.C. and Yu, Y.T. 2008, August. Designing an automatic debugging assistant for improving the learning of computer programming. In International Conference on Hybrid Learning and Education (pp. 359-370). Springer, Berlin, Heidelberg.

Leino, M. 2015. Opeta roboteilla. Lumat 3(7).

Lester, J., Spires, H., Nietfeld, J., Minogue, J., Mott, B. & Lobene, E. 2014. Designing game-based learning environments for elementary science education: A narrative-centered learning perspective. Information Sciences 264.

Lister, R., Fidge, C. and Teague, D. 2009. Further evidence of a relationship between explaining, tracing and writing skills in introductory programming. Acm sigcse bulletin, 41(3), pp.161-165.

Ma, H. 2013. Tech Services on the Web: Codecademy; http://www. codecademy. com. Technical Services Quarterly, 30(3), 338-339.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., and Eastmond, E. 2010. The scratch programming language and environment. ACM Trans. Comput. Educ. 10, 4, Article 16

Mannila, L., de Raadt, M. 2006. An Objective Comparison of Languages for Teaching Introductory Programming. Proceedings, Koli Calling.

Mardis, M., Everhart, N., Smith, D., Newsum, J., Baker, S. 2010. From Paper to Pixel: Digital Textbooks and Florida's Schools. Florida State University Libraries.

Markus B. 2008. Thinking about e-Learning, FIG workshop, ITC, Enschede.

Marttala, L. 2017. Tietotekniikan valtakunnallisten oppisisältöjen toteutuminen Keski-Suomen peruskoulujen opetuskäytänteissä. Jyväskylän yliopisto.

Meisalo V., Sutinen E. & Tarhio J. 2003. Modernit oppimisympäristöt – Tieto- ja viestintätekniikka opetuksen ja opiskelun tukena, Tietosanoma.

Meisalo, V., Sutinen, E. & Tarhio, J. 2000. Modernit oppimisympäristöt. Helsinki: Tietosanoma.

Mészárosová, E. 2015. Is Python an Appropriate Programming Language for Teaching Programming in Secondary Schools? International Journal of Information and Communication Technologies in Education, 4(2), 5-14.

Microbit, www.microbit.org/fi/guide. Katsottu 2.12. 2019.

Mykkänen, J. & Liukas, L. 2014. Koodi 2016. Helsinki: Lönnberg Print.

Oddie, A., Hazlewood, P., Blakeway, S. and Whitfield, A. 2010. Introductory problem solving and programming: Robotics versus traditional approaches. Innovation in Teaching and Learning in Information and Computer Sciences, 9(2), pp.1-11.

Opetushallitus. 2014. Perusopetuksen opetussuunnitelman perusteet. Määräykset ja ohjeet 2014: 96.

Orfanakis, V., Papadakis, S. 2016. Teaching basic programming concepts to novice programmers in Secondary Education using Twitter, Python, Ardruino and a coffee machine. Hellenic Conference on Innovating STEM Education (HISTEM), 16–18 December 2016, University of Athens, Greece.

Ornelas Marques, F., Marques, M.T. 2012. No problem? No research, little learning ... big problem!. Systemics, Cybernetics and Informatics, 10(3), 60–62.

Otava, 2020a oppimisenpalvelut.otava.fi/tuotteet/luokat-7-9/pii-ops-2016/. Katsottu: 9.1.2020.

Otava, 2020b. https://otavakonserni.fi/. Katsottu: 17.2.2020

Papadakis, S. 2018 ‘The use of computer games in classroom environment’, Int. J. Teaching and Case Studies, Vol. 9, No. 1, pp.1–25.

Papadakis, St., Kalogiannakis, M., Orfanakis, V., & Zaranis, N. 2015. Novice Programming Environments. Scratch & App Inventor: a first comparison. In Proceedings of the 2014 Workshop on Interaction Design in Educational Environments (IDEE '14). ACM, New York, NY, USA.

Paulsson, F., & Naeve, A. 2003. Standardized content archive management – SCAM – storing and distributing learning resources. IEEE Learning Technology Newsletter, Vol 5.

Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., Devlin, M. and Paterson, J. 2007, December. A survey of literature on the teaching of introductory programming. In ACM sigcse bulletin (Vol. 39, No. 4, pp. 204-223). ACM.

Perkkilä, P., Joutsenlahti, J., Sarenius, V.-M. 2018. Peruskoulun matematiikan oppikirjat osana oppimateriaalitutkimusta. Matematiikan opetus ja oppiminen, Niilo Mäki Instituutti.

Perrotta, C., Featherstone, G., Aston, H. & Houghton, E. 2013. Game-based Learning: Latest Evidence and Future Directions (NFER Research Programme: Innovation in Education). Slough: NFER.

Pihola, P. 2016. ASIMOV Ensimmäiset robottini. Innokas-verkosto.

Prensky, M. 2005 ‘Computer games and learning: digital game-based learning’, in Joost, R. and Goldstein, J. (Eds.): Handbook of Computer Game Studies, pp.97–122, The MIT Press, Cambridge, London.

Reas, C. and Fry, B. 2006. Processing: programming for the media arts. AI & SOCIETY, 20(4), pp.526-538.

Richmond, S., 2013. Wearable computing is here already: How hi-tech got under our skin. The Independent.

Robbo. www.robbo.world. Katsottu 14.1.2020.

Robins, A., Rountree, J., Rountree, N. 2003. Learning and Teaching Programming: A Review and Discussion. Computer Science Education Vol. 13.

Rongas, T., Kaarna, A. and Kalviainen, H. 2004. Classification of computerized learning tools for introductory programming courses: learning approach. In IEEE International Conference on Advanced Learning Technologies, 2004. Proceedings. (pp. 678-680). IEEE.

Rusk, N., Resnick, M., Berg, R., & Pezalla-Granlund, M. 2008. New pathways into robotics: strategies for broadening participation. Journal of Science Education and Technology, 17(1), 59–69.

Sanner M.F. 1999. Python: a programming language for software integration and development. J Mol Graph Model, 17(1), pp.57-61.

Sanoma Pro. 2020a, www.sanomapro.fi/sarjat/kuutio/. Katsottu: 9.1.2020.

Sanoma Pro. 2020b, www.sanomapro.fi/sarjat/muuttuja/. Katsottu: 9.1.2020.

Sanoma Pro, 2020c. https://www.sanomapro.fi/tietoa-meista/

Selby, C.C. 2015. Relationships: computational thinking, pedagogy of programming, and Bloom's Taxonomy. In Proceedings of the Workshop in Primary and Secondary Computing Education (WiPSCE '15). ACM, New York, NY, USA, 80-87.

Selby, C.C., Woollard, J. 2013. Computational Thinking: The Developing Definition.

Sentance, S., Waite, J., Hodges, S., MacLeod, E., Yeomans, L. 2017. Creating Cool Stuff: Pupils' Experience of the BBC micro: bit, in: Proceedings 2017 ACM SIGCSE Technical SymposiumComputerScienceEducation. ACM, pp. 531–536.

Sentance, S., Waite, J. and Kallia, M. 2019. Teachers' Experiences of using PRIMM to Teach Programming in School. In Proceedings of the 50th ACM Technical Symposium on Computer Science Education (pp. 476-482). ACM.

Spalter, A. M., & van Dam, A. 2003. Problems with using components in educational software. Computers & Graphics, 27.

Studeo, 2020. www.studeo.fi/. Katsottu: 19.2.2020

Sykora, C. 2014. Computational thinking for all. www.iste.org/explore/articleDetail?articleid=152, luettu 16.1.2020

Tsukamoto, H., Takemura, Y., Nagumo, H., Ikeda, I., Monden, A. and Matsumoto, K.I. 2015. Programming education for primary school children using a textual programming language. In 2015 IEEE Frontiers in Education Conference (FIE) (pp. 1-7). IEEE.

Turun yliopisto, Oppimisanalytiikan keskus, 2020. www.oppimisanalytiikka.fi/2019/toiminta#projektit. Katsottu: 18.2.2020

Uotila, K. 2016. Ohjelmointi yläasteen matematiikassa, Helsingin yliopisto.

Vavoula, G., Sharples, M., Rudman, P., Lonsdale, P. & Meek, J. 2007. Learning bridges: A role for mobile technologies in education. Educational Technology XLVII.

Verdú, E., Regueras, L.M., Verdú, M.J., Leal, J.P., de Castro, J.P. and Queirós, R. 2012. A distributed system for learning programming on-line. Computers & Education, 58(1), pp.1-10.

Wagner, A., Gray, J., Corley, J. and Wolber, D. 2013. Using app inventor in a K-12 summer camp. In Proceeding of the 44th ACM technical symposium on Computer science education (pp. 621-626). ACM.

Williams L., Wiebe E., Yang K., Ferzli M., Miller C. 2002. In Support of Pair Programming in the Introductory Computer Science Course, Computer Science Education

Wing, J.M. 2008. Computational thinking and thinking about computing. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 366(1881), pp.3717-3725.

Wulf, T. 2005. Constructivist approaches for teaching computer programming. In SIGITE'05, October 20–22, Newark, Jew Jersey, USA, pp. 245–248.

Yahya, S., Ahmad, E. A., Jalil, K. A. 2010. The definition and characteristics of ubiquitous learning: A discussion. International Journal of Education and Development using Information and Communication Technology (IJEDICT), Vol. 6, Issue 1.

Yau, J., Joy, M. 2006. Context-aware and adaptive learning schedule for mobile learning. International Workshop on Mobile and Ubiquitous Learning Environments (MULE) at the International Conference on Computers in Education (ICCE 2006), Beijing, China.

Liitteet

Liite 1. Kyselyssä olleet kysymykset.

TAUSTATIETOA

1. Missä koulussa toimit opettajana? (koulun nimi ja paikkakunta)

2. Kuinka monta tuntia matematiikkaa opetat viikossa?

3. Opetatko ohjelmointia lukuvuonna 2017-2018?

- Kyllä
- Ei

4. Jos et, mitkä ovat suurimmat syyt/esteet siihen?

5. Oletko ohjelmoinut aikaisemmin (esim. Scratch, Java, Lego-robotit)

OHJELMOINNIN OPETUS KÄYTÄNNÖSSÄ

6. Mitä ohjelmointikieltä/-kieliä käytät/aiot käyttää opetuksessasi?

7. Mitä ohjelmointiympäristöjä käytät/aiot käyttää opetuksessa?

- ☐ Tietokoneelle asennettu ohjelmisto, esim. Netbeans, Kompozer
- ☐ Internet-ympäristö, esim. Scratch, Hour of Code
- ☐ Muu, mikä?

8. Kerro tarkemmin, mitä ohjelmointiympäristöä olet käyttänyt ja onko se mielestäsi toiminut opetuksessa hyvin.

KOULUN TEKNOLOGIAYMPÄRISTÖ

9. Millaiset laitteet koulullasi on käytössä ohjelmoinnin opetusta varten?

- ☐ atk-luokka
- ☐ oppilaiden omat kannettavat
- ☐ ohjelmoitavan robotti
- ☐ älytaulu (smart board)
- ☐ ei mitään
- ☐ Muu, mikä?

10. Ovatko koululla käytettävät laitteet/tarvikkeet riittäviä ohjelmoinnin opetukseen?

- Kyllä
- Ei

11. Jos eivät, mitä voisi parantaa laitteiden osalta?

OPPIMATERIAALIT

12. Mitä matematiikan oppimateriaalia käytät opetuksessasi?

13. Käytätkö perinteistä oppikirjaa, digikirjaa, molempia vai jotain muuta?

- Perinteinen oppikirja
- Digikirja
- Käytän molempia
- Muu, mikä?

14. Onko ohjelmointi mielestäsi sisällytetty hyvin kyseiseen oppimateriaaliin?

- Kyllä
- Ei

15. Käytätkö muuta materiaalia ohjelmoinnin opetukseen? Jos käytät, niin mitä?

16. Tukevatko ohjelmoinnin oppimateriaalit opetussuunnitelmaan asetettujen tavoitteiden saavuttamista? *Oppilas osaa soveltaa algoritmisen ajattelun periaatteita ja osaa ohjelmoida yksinkertaisia ohjelmia.*

- Kyllä
- Ei

17. Miten ohjelmointi on sisällytetty kuntakohtaiseen opetussuunnitelmaan?

18. Onko ohjelmoinnin oppimateriaaleissa mielestäsi puutteita? Jos on mitä?

19. Minkä tyyppisiä oppimateriaaleja toivoisit ohjelmoinnin opetukseen?

KOKEMUKSIA OHJELMOINNIN OPETUKSESTA

20. Minkälaisia haasteita olet kohdannut ohjelmoinnin opetuksessa?